

Renesas Embedded
Application Programming Interface
Reference manual

Reference Manual

Rev.1.01
Feb 16, 2007

REVISION HISTORY

Rev.	Summary	Date
1.00		06.04.07
1.01	<ul style="list-style-type: none"> • Additions and corrections made for support of libraries for the H8/36094, H8/36077, H8/36109, and R8C/24 and 25 • Unnecessary periods in Chapter 1, “Instruction,” deleted • Port P73 of H8 deleted • Channels A–D written in CreateInputCapture corrected to input captures A–D • Error in writing “seroal” corrected to “serial” • Erroneous description of RAPI_70_STATES corrected • Error in writing of timer channel names corrected • Error in writing of input capture mode corrected • Error in writing of output compare mode corrected • Erroneous description of RAPI_OVERFLOW_BIT15 corrected • Support for R8C UART1 clock synchronous mode • RAPI_COUNT_CLEAR added to first arguments for the R8C and H8/300H in __CreateInputCapture and __CreateOutputCompare • “External signal” deleted from the timer RD count source for the R8C in __CreateInputCapture and __CreateOutputCompare • Following statements added to the item for timer RD in __SetTimerRegister and __GetTimerRegister • [15]: Timer RD output master enable register 1 • [16]: Timer RD output master enable register 2 • [17]: Timer RD output control register • Explanation for clearing H8/300H module added to the description of __Create..., __Open..., and __BasicOpen... • Explanation for setting H8/300H module for standby added to the description of __Destroy..., __Close..., and BasicClose... • Return value in __PollingSerialReceiving changed from Boolean to unsigned int, and explanation of return value and example program changed • Return value in __PollingSerialSending changed from Boolean to unsigned int, and explanation of return value and example program changed • Remarks regarding use of port B on H8/300H added to __ReadIOPort and __ReadIOPortRegister 	07.02.16

	<ul style="list-style-type: none"> • Description of third argument in __CreateInputCapture corrected • Wording “specified” unified to “specified” • RAPI_BOTH deleted from the description of timer B1 count edges of H8/300H in __CreateEventCounter • RAPI_FTIOA and RAPI_FTIOB added to __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode • All occurrences of the word “Tiny” deleted • Description of second argument added to __GetPulsePeriodMeasurementMode, __GetPulseWidthMeasurementMode, and __GetEventCounter • Description regarding timer V trigger of H8 in __CreateTimer, __CreateEventCounter, and __CreatePulseWidthMeasurementMode corrected • Explanation of H8 interrupt settings and interrupt control register settings in __SetSerialInterrupt corrected • Remarks on timer V of H8/300H in __EnableTimerRegister added • Specification of timer RA input pins of R8C in __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode as well as RAPI_TIOSEL_P1_7 and RAPI_TIOSEL_P1_5 added • File name in program example corrected • Item “Reference” in __BasicSetSerialFormat deleted • __BasicSetSerialFormat added to “Reference” in __SetSerialFormat and __SetSerialInterrupt • Error in writing of __CreateInput Capture and __CreateOutputCapture corrected • Explanation added to Section 2.1, “Overview” • Causes of clearing of timer W and timer RC counters in __CreateInputCapture deleted • RAPI_COMPARE_MATCH_A_STOP and RAPI_STOP added to the item for the R8C in __CreateOutputCompare • Changed to RAPI_INT_LV_0 and RAPI_INT_LV_1 for H8/300H in __SetSerialInterrupt • Description relating to clock in __CreateTimer corrected • RAPI_TRC_FILTER and RAPI_TRD_FILTER added to filter specification for H8 in __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode • RAPI_TIMER_RD2 and RAPI_TIMER_RD3 added to timer RD of H8/36109 • Description relating to timer V trigger input for H8/300H in __CreateTimer deleted • Description of pulse output function added to the item for R8C in __CreateEventCounter • Description of timer RE usage added to items data3 and data5 for R8C in __CreateOutputCompare • Filter function of timer RC and timer RD in __CreatePulsePeriodMeasurementMode, __CreatePulseWidthMeasurementMode, and __CreateInputCapture corrected • Description that multiple defined values can be set for data1 in __EnableInterrupt added • Description that RAPI_WITHOUT_SAMPLE_HOLD is specifiable in delay trigger modes 0 and 1 of M16C in __CreateADC corrected • Description that RAPI_FOCOF is specifiable in repeat mode of R8C in __CreateADC corrected 	
--	---	--

	<ul style="list-style-type: none"> • RAPI_AN30, RAPI_AN31, RAPI_AN32, and RAPI_P9_GROUP added to the item for data1 of M16C in __CreateADC • RAPI_AN30, RAPI_AN31, RAPI_AN32, and RAPI_P9_GROUP added to the item for data1 of M16C in __EnableADC • Error in writing of H8/300H interrupt set values in __CreateOutputCompare corrected • Error in writing of timer RD and timer RD symbol name corrected • Description of gate function of M16C in __CreateEventCounter deleted • Description of H8/300H noise rejection function in __BasicSetSerialFormat added • Description of timer W input pins of H8/300H in __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode deleted 	
--	---	--

Table of Contents

Table of Contents	5
1. Introduction	8
2. Driver	9
2.1 Overview	9
2.2 Driver Features	10
2.3 Serial Interface Driver	11
2.4 Timer Driver	12
2.4.1 Timer Mode	12
2.4.2 Event Counter Mode	12
2.4.3 Pulse Width Modulation Mode (PWM Mode)	12
2.4.4 Pulse Period Measurement Mode	12
2.4.5 Pulse Width Measurement Mode	12
2.4.6 Input Capture Mode	12
2.4.7 Output Compare Mode	13
2.5 I/O Port Driver	14
2.6 External Interrupt Driver	15
2.7 A/D Converter Driver	16
3. Standard Types	18
4. Library Reference	19
4.1 API List by Peripheral Facility	19
4.2 Description of Each API	21
4.2.1 Serial I/O	22
__BasicOpenSerialDriver	22
__BasicCloseSerialDriver	23
__BasicSetSerialFormat	24
__BasicStartSerialReceiving	29
__BasicStartSerialSending	30
__BasicReceivingStatusRead	31
__BasicSendingStatusRead	33
__BasicStopSerialReceiving	34
__BasicStopSerialSending	35
__OpenSerialDriver	36
__CloseSerialDriver	37
__ConfigSerialDriverNotify	38
__SetSerialFormat	41
__SetSerialInterrupt	42
__StartSerialReceiving	45
__StartSerialSending	47
__StopSerialReceiving	49
__StopSerialSending	50
__PollingSerialReceiving	51
__PollingSerialSending	52
4.2.2 Timer	54

__CreateTimer	54
__EnableTimer	60
__DestroyTimer	62
__CreateEventCounter	64
__EnableEventCounter	69
__DestroyEventCounter	71
__GetEventCounter	73
__CreatePulseWidthModulationMode	75
__EnablePulseWidthModulationMode	80
__DestroyPulseWidthModulationMode	82
__CreatePulsePeriodMeasurementMode	83
__EnablePulsePeriodMeasurementMode	88
__DestroyPulsePeriodMeasurementMode	90
__GetPulsePeriodMeasurementMode	92
__CreatePulseWidthMeasurementMode	94
__EnablePulseWidthMeasurementMode	99
__DestroyPulseWidthMeasurementMode	101
__GetPulseWidthMeasurementMode	103
__CreateInputCapture	105
__EnableInputCapture	115
__DestroyInputCapture	117
__GetInputCapture	119
__CreateOutputCompare	121
__EnableOutputCompare	133
__DestroyOutputCompare	135
__SetTimerRegister	137
__EnableTimerRegister	143
__ClearTimerRegister	145
__GetTimerRegister	147
4.2.3 I/O Port	153
__SetIOPort	153
__ReadIOPort	158
__WriteIOPort	162
__SetIOPortRegister	166
__ReadIOPortRegister	168
__WriteIOPortRegister	170
4.2.4 External interrupt	172
__SetInterrupt	172
__EnableInterrupt	176
__GetInterruptFlag	178
__ClearInterruptFlag	180
4.2.5 A/D converter	182
__CreateADC	182
__EnableADC	191
__DestroyADC	196

__GetADC	197
__GetADCAI1	198
__GetADCStatus	199
__ClearADCStatus	200

1. Introduction

The Renesas Embedded Application Programming Interface (API) is a unified API for the microcomputers made by Renesas Technology Corporation.

2. Driver

2.1 Overview

The library described herein provides a peripheral facility control program (peripheral driver) for microcomputers. Use of the Renesas API permits the peripheral driver to be built into a user program.

Configuration of Renesas APIs is shown below.

File name	Description
rapi_xxx_yyy.lib (xxx = family name, yyy = series name)	This is the Renesas API library file. To use Renesas APIs, specify this file as input file for the linker.
rapi_ad_xxx_yyy.h (xxx = family name, yyy = series name)	This is the header file for A/D converter driver APIs. To use A/D converter driver APIs, be sure to include this file.
rapi_io_port_xxx_yyy.h (xxx = family name, yyy = series name)	This is the header file for I/O port driver APIs. To use I/O port driver APIs, be sure to include this file.
rapi_sif_xxx_yyy.h (xxx = family name, yyy = series name)	This is the header file for serial I/F driver APIs. To use serial I/F driver APIs, be sure to include this file.
rapi_timer_xxx_yyy.h (xxx = family name, yyy = series name)	This is the header file for timer driver APIs. To use timer driver APIs, be sure to include this file.
Interrupt_xxx_yyy.h (xxx = family name, yyy = series name)	This is the interrupt function source file for Renesas APIs. To use Renesas APIs that use interrupts, add this file to the user program.
rapi_io_xxx_yyy.h (xxx = family name, yyy = series name)	This is the CPU control register definition header file for Renesas APIs.

2.2 Driver Features

The library described herein has the following features available as a peripheral driver.

(1) Serial I/O control feature

It comprises a serial interface driver, which sets or clears the conditions of serial communication, as well as controls and manages the transmission/reception of communication data.

(2) Timer control feature

It comprises a timer driver, which sets or clears the operating conditions of timers, as well as controls the timer operation.

(3) I/O port control feature

It comprises an I/O port driver, which sets or clears the usage conditions of I/O ports, as well as control data read/write operation.

(4) External interrupt control feature

It comprises an external interrupt driver, which sets or clears the usage conditions of external interrupts, as well as controls interrupt operation.

(5) A/D converter control feature

It comprises an A/D converter driver, which sets or clears the usage conditions of A/D converters, as well as controls A/D converter operation.

2.3 Serial Interface Driver

The serial interface driver sets serial communication, clears settings, transmit/receives data, and controls the status of serial communication.

There are two kinds of serial interface driver: a single-data transmission/reception API and a multi-data transmission/reception API.

2.4 Timer Driver

The timer driver sets the timer, clears timer settings, controls timer operation, and acquires a counter value with respect to the following modes:

- Timer mode
- Event counter mode
- Pulse width modulation mode (PWM mode)
- Pulse period measurement mode
- Pulse width measurement mode
- Input capture mode
- Output compare mode

2.4.1 Timer Mode

In this mode, the timer counts the internally generated count source. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.2 Event Counter Mode

In this mode, the timer counts the external signal fed in from an input pin or an overflow or underflow from other timer. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.3 Pulse Width Modulation Mode (PWM Mode)

In this mode, the timer outputs pulses in a given width successively. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.4 Pulse Period Measurement Mode

In this mode, the timer measures the pulse period of an external signal fed in from an input pin. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.5 Pulse Width Measurement Mode

In this mode, the timer measures the pulse width of an external signal fed in from an input pin. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.6 Input Capture Mode

In this mode, the timer latches the timer value upon an active signal edge or clock pulse at an input pin, thereby generating an interrupt request. When an input capture interrupt or an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.7 Output Compare Mode

In this mode, the timer generates an interrupt request when the timer counter and a comparison value match. When a compare match interrupt or an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.5 I/O Port Driver

The I/O port driver sets the I/O port for input or output, writes data to the I/O port, and reads data from the I/O port.

2.6 External Interrupt Driver

The external interrupt driver sets external interrupts, controls external interrupts, acquires the status of external interrupt flags, and clears external interrupt flags.

2.7 A/D Converter Driver

The A/D converter driver sets the A/D converter, controls the A/D converter, clears settings of the A/D converter, acquires the A/D converter value, acquires the status of the A/D converter, and clears the status of the A/D converter.

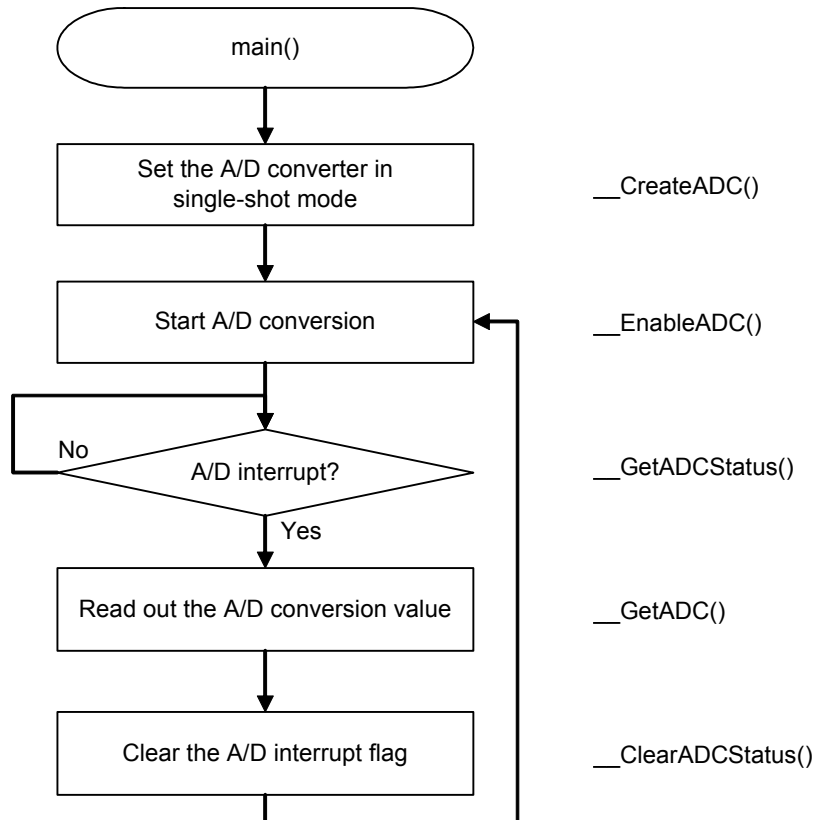
[A/D converter driver usage example (single-shot mode)]

Here, a program example is shown for the A/D converter operating in single-shot mode under the conditions given below.

CPU	:	R8C
Operating clock	:	fAD divided by 2
Resolution	:	10 bits
Analog input pin	:	AN0 pin
A/D conversion start condition	:	Software trigger
Sample-and-hold	:	Enabled

A program flow and a program example are shown below.

(Program flow)



Program example

```
#include "rapi_ad_r8c_13.h"

void main( void )
{
    unsigned int  status, data;

    /* Set up A/D converter as one short mode */
    __CreateADC(          RAPI_ONE_SHOT|RAPI_AN0|RAPI_FAD2|
RAPI_WITH_SAMPLE_HOLD|
                    RAPI_AD_OFF|RAPI_10BIT, 1, 0, 0 );

    while( 1 ){
        /* Disable A/D converter */
        __EnableADC( RAPI_AN0| RAPI_AD_ON, 1 );

        /* Check a flag bit of A/D converter interrupt */
        do{
            __GetADCStatus( &status );
        } while( (*status & 0x0001) == 0 )

        /* Get A/D converted data of A/D register */
        __GetADC( &data );

        /* Clear status of A/D converted */
        __ClearADCStatus( 0 );
    }
}
```

3. Standard Types

This section describes the standard types defined in the library. For details about the set values, refer to the description of each API.

Standard type	Description
Boolean	The Boolean type represents the enum-type data that indicates whether successful (RAPI_TRUE (= 1)) or failed (RAPI_FALSE (= 0)).
VoidFuncNotify	The VoidFuncNotify type represents the type of the notification function to be registered.

4. Library Reference

4.1 API List by Peripheral Facility

The table below lists the Renesas Embedded APIs classified by peripheral facility.

NO	Facility classification	API	API operation
1	Single-data serial I/O	__BasicOpenSerialDriver	Opens serial port
2		__BasicCloseSerialDriver	Closes serial port
3		__BasicSetSerialFormat	Sets serial communication
4		__BasicStartSerialReceiving	Receives 1 data
5		__BasicStartSerialSending	Transmits 1 data
6		__BasicReceivingStatusRead	Reads receive status
7		__BasicSendingStatusRead	Reads transmit status
8		__BasicStopSerialReceiving	Stops reception
9		__BasicStopSerialSending	Stops transmission
10	Multi-data serial I/O	__OpenSerialDriver	Opens serial port
11		__CloseSerialDriver	Closes serial port
12		__ConfigSerialDriverNotify	Registers notification function
13		__SetSerialFormat	Sets serial communication
14		__SetSerialInterrupt	Sets transmit/receive interrupt
15		__StartSerialReceiving	Starts reception
16		__StartSerialSending	Starts transmission
17		__StopSerialReceiving	Stops reception
18		__StopSerialSending	Stops transmission
19		__PollingSerialReceiving	Receives by polling
20	__PollingSerialSending	Transmits by polling	
21	Timer	__CreateTimer	Sets timer mode
22		__EnableTimer	Controls timer mode operation
23		__DestroyTimer	Clears timer mode setting
24		__CreateEventCounter	Sets event counter mode
25		__EnableEventCounter	Controls operation of event counter mode
26		__DestroyEventCounter	Clears setting of event counter mode
27		__GetEventCounter	Gets event counter mode counter value
28		__CreatePulseWidthModulationMode	Sets pulse width modulation mode
29		__EnablePulseWidthModulationMode	Controls operation of pulse width modulation mode
30		__DestroyPulseWidthModulationMode	Clears setting of pulse width modulation mode
31		__CreatePulsePeriodMeasurementMode	Sets pulse period measurement mode
32		__EnablePulsePeriodMeasurementMode	Controls operation of pulse period measurement mode
33		__DestroyPulsePeriodMeasurementMode	Clears setting of pulse width measurement mode
34		__GetPulsePeriodMeasurementMode	Acquires measured value of pulse period measurement mode

35		__CreatePulseWidthMeasurementMode	Sets pulse width measurement mode
36		__EnablePulseWidthMeasurementMode	Controls operation of pulse width measurement mode
37	Timer	__DestroyPulseWidthMeasurementMode	Clears setting of pulse width measurement mode
38		__GetPulseWidthMeasurementMode	Acquires measured value of pulse width measurement mode
39		__CreateInputCapture	Sets input capture mode
40		__EnableInputCapture	Controls operation of input capture mode
41		__DestroyInputCapture	Clears setting of input capture mode
42		__GetInputCapture	Acquires counter value of input capture mode
43		__CreateOutputCompare	Sets output compare mode
44		__EnableOutputCompare	Controls operation of output compare mode
45		__DestroyOutputCompare	Clears setting of output compare mode
46		__SetTimerRegister	Sets timer register
47		__EnableTimerRegister	Controls operation of timer register
48		__ClearTimerRegister	Clears timer register
49		__GetTimerRegister	Gets timer register value
50	I/O port	__SetIOPort	Sets I/O port
51		__ReadIOPort	Reads from I/O port
52		__WriteIOPort	Writes to I/O port
53		__SetIOPortRegister	Sets I/O port register
54		__ReadIOPortRegister	Reads from I/O port register
55		__WriteIOPortRegister	Writes to I/O port register
56	External interrupt	__SetInterrupt	Sets external interrupt
57		__EnableInterrupt	Controls external interrupt
58		__GetInterruptFlag	Gets flag status of external interrupt
59		__ClearInterruptFlag	Clears flag of external interrupt
60	A/D converter	__CreateADC	Sets A/D converter
61		__EnableADC	Controls operation of A/D converter
62		__DestroyADC	Discards settings of A/D converter
63		__GetADC	Gets A/D conversion value (register specified)
64		__GetADCAI	Gets A/D conversion value (all registers)
65		__GetADCStatus	Gets status of A/D converter
66		__ClearADCStatus	Clears status of A/D converter

4.2 Description of Each API

This section describes each API and explains how to use them, showing a program example for each.

The description of each API is divided into the following items.

- **Synopsis** : Outlines the content of processing performed by the function. It also shows the syntax of the function, followed by a brief explanation of arguments.
- **Description** : Describes the function and how to use it in detail.
- **Return value** : Explains the returned value of the function.
- **Functionality** : Indicates the functional classification of the function.
- **Reference** : Indicates the related functions.
- **Remark** : Describes the precautions to be taken when using the API.
- **Program example** : Presents a program showing how to use the function.

4.2.1 Serial I/O

__BasicOpenSerialDriver

Synopsis

<Open a serial port>

Boolean __BasicOpenSerialDriver(unsigned long data)

data	Setup data
------	------------

Description

Opens and initializes a specified serial port.

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Reference

[__BasicCloseSerialDriver](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8/300H, this API opens and initializes a specified serial port when freeing it from module standby state.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    /* Open serial driver */
    return __BasicOpenSerialDriver( RAPI_COM1 );
}
```

__BasicCloseSerialDriver

Synopsis

<Close a serial port>

Boolean **__BasicCloseSerialDriver(unsigned long data)**

data	Setup data
------	------------

Description

Closes a specified serial port. For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Reference

[__BasicOpenSerialDriver](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8/300H, this API places a specified serial port into module standby state after closing it.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    /* Close serial driver */
    return Rapi_BasicCloseSerialDriver( RAPI_COM1 );
}
```

__BasicSetSerialFormat

Synopsis

<Set serial communication>

Boolean __BasicSetSerialFormat(unsigned long data1, unsigned char data2)

data1	Setup data 1
data2	Setup data 2

Description

Sets serial communication according to specified parameters.

[data1]

For data1, the following values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

For serial communication mode, the following values can be set.

(M16C) (UART0, UART1, UART2)

RAPI_SM_SYNC	Clock synchronous serial communication mode
RAPI_SM_ASYNC	Clock asynchronous serial communication mode

(M16C)(SI/O3, SI/O4)

RAPI_SIO_SM_SYNC	Clock synchronous serial communication mode
------------------	---

For the data length format of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_BIT_7	Transfer data length 7 bits	RAPI_BIT_8	Transfer data length 8 bits
RAPI_BIT_9	Transfer data length 9 bits		

For the clock source of serial communication, the following values can be set.

(M16C) (UART0, UART1, UART2)

RAPI_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_CKDIR_EXT	External clock is used as the clock source of serial communication.

(M16C) (SI/O3, SI/O4)

RAPI_SIO_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_SIO_CKDIR_EXT	External clock is used as the clock source of serial communication.

For the stop bit length of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_STPB_1	1 stop bit	RAPI_STPB_2	2 stop bits
-------------	------------	-------------	-------------

For the parity bit of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_PARITY_NON	No parity bit	RAPI_PARITY_EVEN	Even parity bit
RAPI_PARITY_ODD	Odd parity bit		

For the clock polarity of serial communication, the following values can be set.

If the API is used in clock asynchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_DPOL_NON	Polarity not inverted	RAPI_DPOL_INV	Polarity inverted
---------------	-----------------------	---------------	-------------------

(M16C) (SI/O3, SI/O4)

RAPI_SIO_DPOL_NON	Polarity not inverted	RAPI_SIO_DPOL_INV	Polarity inverted
-------------------	-----------------------	-------------------	-------------------

For the count source of the built-in baud rate generator, the following values can be set.

(M16C) (UART0, UART1, UART2)

RAPI_BCSS_F1	f1SIO	RAPI_BCSS_F2	f2SIO
RAPI_BCSS_F8	f8SIO	RAPI_BCSS_F32	f32SIO

(M16C) (SI/O3, SI/O4)

RAPI_SIO_BCSS_F1	f1SIO	RAPI_SIO_BCSS_F2	f2SIO
RAPI_SIO_BCSS_F8	f8SIO	RAPI_SIO_BCSS_F32	f32SIO

For the _CTS/_RTS function, the following values can be set.

If the internal clock is selected for use in clock synchronous serial communication mode, the _RTS function has no effect.

(M16C) (UART0, UART1, UART2)

RAPI_CTSRTS_DIS	_CTS/_RTS functions are not used.
RAPI_CTS_SEL	_CTS function is selected.
RAPI_RTS_SEL	_RTS function is selected.

For the transfer format, the following values can be set.

If the data length selected for use in clock asynchronous serial communication mode is 7 or 9 bits long, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_LSB_SEL	LSB first	RAPI_MSB_SEL	MSB first
--------------	-----------	--------------	-----------

(M16C) (SI/O3, SI/O4)

RAPI_SIO_LSB_SEL	LSB first	RAPI_SIO_MSB_SEL	MSB first
------------------	-----------	------------------	-----------

For serial data logic switchover, the following values can be set.

(M16C) (UART2)

RAPI_LOGIC_NO_REV	The value written in the transmit buffer register does not have its logic inverted.
RAPI_LOGIC_REV	The value written in the transmit buffer register is inverted before being transmitted.

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

For serial communication mode, the following values can be set.

(R8C) (UART0, UART1)

RAPI_SM_SYNC	Clock synchronous serial communication mode
RAPI_SM_ASYNC	Clock asynchronous serial communication mode

For the data length format of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1)

RAPI_BIT_7	Transfer data length 7 bits	RAPI_BIT_8	Transfer data length 8 bits
RAPI_BIT_9	Transfer data length 9 bits		

For the clock source of serial communication, the following values can be set.

(R8C) (UART0, UART1)

RAPI_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_CKDIR_EXT	External clock is used as the clock source of serial communication.

For the stop bit length of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1)

RAPI_STPB_1	1 stop bit	RAPI_STPB_2	2 stop bits
-------------	------------	-------------	-------------

For the parity bit of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1)

RAPI_PARITY_NON	No parity bit	RAPI_PARITY_EVEN	Even parity bit
RAPI_PARITY_ODD	Odd parity bit		

For the clock polarity of serial communication, the following values can be set.

If the API is used in clock asynchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1)

RAPI_DPOL_NON	Polarity not inverted	RAPI_DPOL_INV	Polarity inverted
---------------	-----------------------	---------------	-------------------

For the count source of the built-in baud rate generator, the following values can be set.

(R8C) (UART0, UART1)

RAPI_BCSS_F1	f1SIO	RAPI_BCSS_F8	f8SIO
RAPI_BCSS_F32	f32SIO		

For the transfer format, the following values can be set.

If the data length selected for use in clock asynchronous serial communication mode is 7 or 9 bits long, do not set these values.

(R8C) (UART0, UART1)

RAPI_LSB_SEL	LSB first	RAPI_MSB_SEL	MSB first
--------------	-----------	--------------	-----------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
-----------	----------------	-----------	----------------

RAPI_COM3	SCI3 channel 3
-----------	----------------

For serial communication mode, the following values can be set.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_SM_SYNC	Clock synchronous serial communication mode
RAPI_SM_ASYNC	Clock asynchronous serial communication mode

For the data length format of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_BIT_7	Transfer data length 7 bits	RAPI_BIT_8	Transfer data length 8 bits
------------	-----------------------------	------------	-----------------------------

For the clock source of serial communication, the following values can be set.

(H8/300H) (SCI3 channel 1, SCI3 channel 2)

RAPI_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_CKDIR_EXT	External clock is used as the clock source of serial communication.

For the stop bit length of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_STPB_1	1 stop bit	RAPI_STPB_2	2 stop bits
-------------	------------	-------------	-------------

For the parity bit of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_PARITY_NON	No parity bit	RAPI_PARITY_EVEN	Even parity bit
RAPI_PARITY_ODD	Odd parity bit		

For the count source of the built-in baud rate generator, the following values can be set.

(H8/300H) (SCI3 channel 1, SCI3 channel 2 SCI3 channel 3)

RAPI_BCSS_F1	Φ clock	RAPI_BCSS_F4	$\phi/4$ clock
RAPI_BCSS_F16	$\phi/16$ clock	RAPI_BCSS_F32	$\phi/64$ clock

The noise rejection function can be set to be turned on or turned off.

When used in clock synchronous serial communication mode, the noise rejection function has no effect.

(H8/300H) (SCI3 channel 3)

RAPI_NOISE_CANCEL_ON	Noise rejection function turned on
RAPI_NOISE_CANCEL_OFF	Noise rejection function turned off

[data2]

Sets the divide-by-N value of a communication speed.

Return value

If serial communication was successfully set, RAPI_TRUE is returned; if settings failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

Boolean func( void )
{
    /* Set the data of RAPI_COM1 to serial driver */
    return _BasicSetSerialFormat(RAPI_COM1 | RAPI_SM_SYNC | RAPI_CKDIR_INT
|
                                RAPI_BCSS_F1 | RAPI_DPOL_NON | RAPI_LSB_SEL,
20);
}
```

__BasicStartSerialReceiving

Synopsis

<Receive 1 data>

Boolean **__BasicStartSerialReceiving(unsigned long data)**

data	Setup data
------	------------

Description

Starts receiving 1 data of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If data reception in serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__BasicReceivingStatusRead](#), [__BasicStopSerialReceiving](#)

Remark

- For the H8/300H, wait for at least a 1-bit period before calling this API after `__BasicSetSerialFormat` was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    __BasicStartSerialReceiving( RAPI_COM1 );
    .....
}
```

__BasicStartSerialSending

Synopsis

<Transmit 1 data>

Boolean __BasicStartSerialSending(unsigned long data1, unsigned int data2)

data	Setup data
data	Transmit data

Description

Starts sending 1 data of serial communication.

For data1, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SIC3 channel 3		

Return value

If data transmission in serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__BasicSendingStatusRead](#), [__BasicStopSerialSending](#)

Remark

- For the H8/300H, wait for at least a 1-bit period before calling this API after __BasicSetSerialFormat was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

Void func( void )
{
    .....
    __BasicStartSerialSending( RAPI_COM1, 0x00AA );
    .....
}
```

BasicReceivingStatusRead

Synopsis

<Read receive status>

unsigned int BasicReceivingStatusRead(unsigned long data)

data	Setup data
------	------------

Description

Returns the receive status of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

The receive status of serial communication is returned. The returned value is one of the following.

(M16C) (UART0, UART1, UART2)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. The value read from the UARTi receive buffer register (i = 0 to 2).

(M16C) (SI/O3, SI/O4)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. Low-order 8 bits: The value read from the SI/Oi transmit/receive register (i = 3, 4).

(R8C)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. The value read from the UARTi receive buffer register (i = 0, 1).

(H8/300H)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. High-order 8 bits: The value read from the serial status register. Low-order 8 bits: The value read from the receive data register. (Not read if an error occurred.)

Functionality

Serial I/O

Reference

[BasicStartSerialReceiving](#), [BasicStopSerialReceiving](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    unsigned int rcv_data;

    .....
    rcv_data = __BasicReceivingStatusRead( RAPI_COM1 );
    .....
}
```


__BasicSendingStatusRead

Synopsis

<Read transmit status>

Boolean **__BasicSendingStatusRead**(unsigned long data)

data	Setup data
------	------------

Description

Returns the transmit status of serial communication. For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If no data exists in the transmit buffer, RAPI_TRUE is returned; if data exists, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__BasicStartSerialSending](#), [__BasicStopSerialSending](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    if ( __BasicSendingStatusRead( RAPI_COM1 ) == RAPI_TRUE ) {
        /* Transmission completion */
    }
    .....
}
```

__BasicStopSerialReceiving

Synopsis

<Stop reception>

Boolean Rapi_BasicStopSerialReceiving(unsigned long data)

data	Setup data
------	------------

Description

Stops receiving data in serial communication

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If data reception in serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__BasicStartSerialReceiving](#)

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    /* Stop receiving data in serial communication */
    __BasicStopSerialReceiving ( RAPI_COM1 );
    .....
}
```

__BasicStopSerialSending

Synopsis

<Stop transmission>

Boolean **__BasicStopSerialSending(unsigned long data)**

data	Setup data
------	------------

Description

Stops transmitting data in serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
SIO3_COM3	SIO3 channel 3		

Return value

If data transmission in serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__BasicStartSerialSending](#)

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- When operating in clock synchronous serial communication mode, data reception is stopped at the same time by this API.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    /* Stop sending data in serial communication */
    __BasicStopSerialSending ( RAPI_COM1 );
    .....
}
```

__OpenSerialDriver

Synopsis <Open a serial port>
Boolean __OpenSerialDriver(unsigned long data)

data	Setup data
------	------------

Description Opens and initializes a specified serial port.
[data]
 For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
SIO3_COM3	SIO3 channel 3		

Return value If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality Serial I/O

Reference [__CloseSerialDriver](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8/300H, this API opens and initializes a specified serial port when freeing it from module standby state.

Program example

```

#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    /* Open serial driver */
    return __OpenSerialDriver( RAPI_COM1 );
    .....
}

```

__CloseSerialDriver

Synopsis

<Close a serial port>

Boolean **__CloseSerialDriver(unsigned long data)**

data	Setup data
------	------------

Description

Closes a specified serial port.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
SIO3_COM3	SIO3 channel 3		

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Reference

[__OpenSerialDriver](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8/300H, this API places a specified serial port into module standby state after closing it.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    /* Close serial driver */
    return __CloseSerialDriver( RAPI_COM1 );
    .....
}
```

__ConfigSerialDriverNotify

Synopsis

<Register a notification function>

Boolean __ConfigSerialDriverNotify(unsigned long data, VoidFuncNotify *func)

data	Setup data
func	Function pointer to be registered

Description

Registers the notification function necessary to get various transmit/receive information of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

[func]

The function to be registered in func must be supplied to the serial I/O driver by the user.

The serial I/O driver calls the function registered in func.

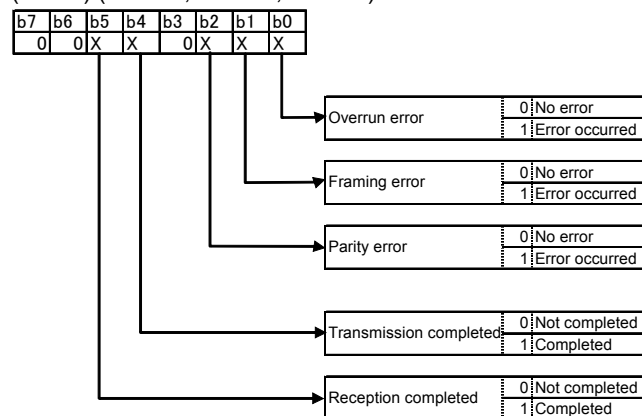
The serial I/O driver notifies the user of the transmit/receive status by an argument.

The type of the function to be registered is shown below.

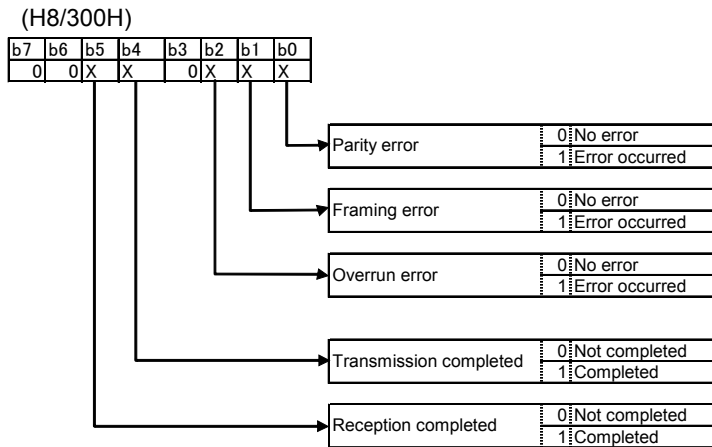
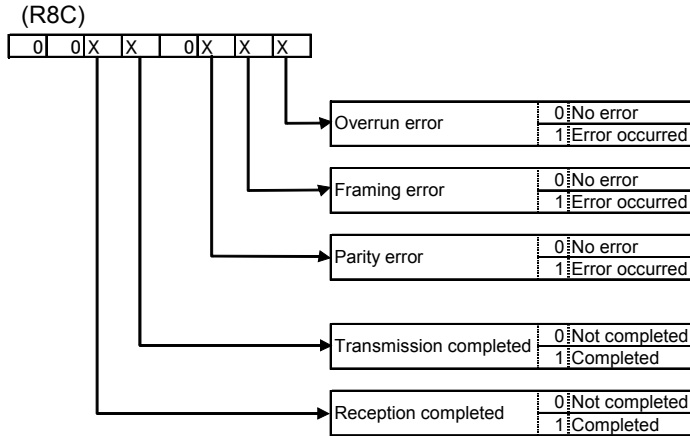
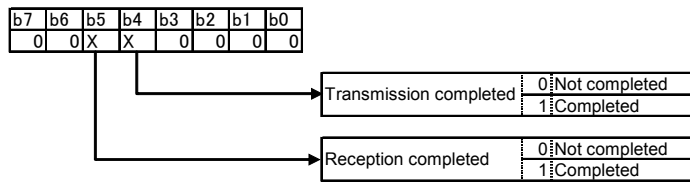
void "any function name" (unsigned char notify);

The argument is detailed below.

(M16C) (UART0, UART1, UART2)



(M16C) (SI/O3, SI/O4)



- | | |
|----------------------|---|
| Return value | If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned. |
| Functionality | Serial I/O |
| Reference | __StartSerialReceiving , __StartSerialSending |
| Remark | <ul style="list-style-type: none"> • The specifiable serial ports differ with each CPU used. • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. |

Program example

```
#include "rapi_sif_r8c_13.h"

void Notify(unsigned char result) {
    if ((result&RAPI_OVER_ERR) == RAPI_OVER_ERR) {
        /* Overrun error */
    }
    if ((result&RAPI_FRAMING_ERR) == RAPI_FRAMING_ERR) {
        /* Framing error */
    }
    if ((result&RAPI_PARITY_ERR) == RAPI_PARITY_ERR) {
        /* Parity error */
    }
    if ((result&RAPI_TX_END) == RAPI_TX_END) {
        /* Transmission completion */
    }
    if ((result&RAPI_RX_END) == RAPI_RX_END) {
        /* Reception completion */
    }
}

Boolean func( void )
{
    .....
    /* Set callback functions of RAPI_COM1 to serial driver */
    return __ConfigSerialDriverNotify( RAPI_COM1, Notify );
    .....
}
```


__SetSerialFormat

Synopsis

<Set serial communication>

Boolean **__SetSerialFormat(unsigned long data1, unsigned char data2)**

data1	Setup data 1
data2	Setup data 2

Description

Sets serial communication according to specified parameters.

For details about parameters, refer to the description of [__BasicSetSerialFormat](#).

Return value

If serial communication was successfully set, RAPI_TRUE is returned; if settings failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__BasicSetSerialFormat](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

Boolean func( void )
{
    .....
    /* Set the data of RAPI_COM1 to serial driver */
    return __SetSerialFormat(RAPI_COM1 | RAPI_SM_SYNC | RAPI_CKDIR_INT |
                             RAPI_BCSS_F1 | RAPI_DPOL_NON | RAPI_LSB_SEL, 20);
    .....
}
```

__SetSerialInterrupt

Synopsis

<Set serial interrupts>

Boolean __SetSerialInterrupt(unsigned long data)

data	Setup data
------	------------

Description

Sets serial interrupts according to specified parameters.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

For interrupt settings, the following values can set.

(M16C) (UART0, UART1, UART2)

RAPI_INT_TX_DIS	Transmit interrupt disabled
RAPI_INT_TX_LV_1	Transmit interrupt priority level 1
RAPI_INT_TX_LV_2	Transmit interrupt priority level 2
RAPI_INT_TX_LV_3	Transmit interrupt priority level 3
RAPI_INT_TX_LV_4	Transmit interrupt priority level 4
RAPI_INT_TX_LV_5	Transmit interrupt priority level 5
RAPI_INT_TX_LV_6	Transmit interrupt priority level 6
RAPI_INT_TX_LV_7	Transmit interrupt priority level 7
RAPI_INT_RX_DIS	Receive interrupt disabled
RAPI_INT_RX_LV_1	Receive interrupt priority level 1
RAPI_INT_RX_LV_2	Receive interrupt priority level 2
RAPI_INT_RX_LV_3	Receive interrupt priority level 3
RAPI_INT_RX_LV_4	Receive interrupt priority level 4
RAPI_INT_RX_LV_5	Receive interrupt priority level 5
RAPI_INT_RX_LV_6	Receive interrupt priority level 6
RAPI_INT_RX_LV_7	Receive interrupt priority level 7

(M16C) (SI/O3, SI/O4)

RAPI_INT_SIO_DIS	SI/O interrupt disabled
RAPI_INT_SIO_LV_1	SI/O interrupt priority level 1
RAPI_INT_SIO_LV_2	SI/O interrupt priority level 2
RAPI_INT_SIO_LV_3	SI/O interrupt priority level 3
RAPI_INT_SIO_LV_4	SI/O interrupt priority level 4
RAPI_INT_SIO_LV_5	SI/O interrupt priority level 5
RAPI_INT_SIO_LV_6	SI/O interrupt priority level 6
RAPI_INT_SIO_LV_7	SI/O interrupt priority level 7

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

For interrupt settings, the following values can set.

RAPI_INT_TX_DIS	Transmit interrupt disabled
RAPI_INT_TX_LV_1	Transmit interrupt priority level 1
RAPI_INT_TX_LV_2	Transmit interrupt priority level 2
RAPI_INT_TX_LV_3	Transmit interrupt priority level 3
RAPI_INT_TX_LV_4	Transmit interrupt priority level 4
RAPI_INT_TX_LV_5	Transmit interrupt priority level 5
RAPI_INT_TX_LV_6	Transmit interrupt priority level 6
RAPI_INT_TX_LV_7	Transmit interrupt priority level 7
RAPI_INT_RX_DIS	Receive interrupt disabled
RAPI_INT_RX_LV_1	Receive interrupt priority level 1
RAPI_INT_RX_LV_2	Receive interrupt priority level 2
RAPI_INT_RX_LV_3	Receive interrupt priority level 3
RAPI_INT_RX_LV_4	Receive interrupt priority level 4
RAPI_INT_RX_LV_5	Receive interrupt priority level 5
RAPI_INT_RX_LV_6	Receive interrupt priority level 6
RAPI_INT_RX_LV_7	Receive interrupt priority level 7

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

For interrupt settings, the following values can set.

RAPI_INT_TX_DIS	Transmit interrupt disabled	RAPI_INT_TX_ENA	Transmit interrupt enabled
RAPI_INT_RX_DIS	Receive interrupt disabled	RAPI_INT_RX_ENA	Receive interrupt enabled

For the CPUs that have an interrupt control register, following values can be set to specify interrupt priority, in addition to ordinary interrupt settings.

RAPI_INT_LV_0	Transmit/Receive interrupt priority level0
RAPI_INT_LV_1	Transmit/Receive interrupt priority level1

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"
```

```
Boolean func( void )
{
    .....
    /* Set interrupt of RAPI_COM1 to serial driver */
    return __SetSerialInterrupt( RAPI_COM1 | RAPI_INT_TX_LV_1 |
RAPI_INT_RX_LV_2 );
    .....
}
```

__StartSerialReceiving

Synopsis

<Start reception>

Boolean __StartSerialReceiving(unsigned long data, unsigned char wordNum, unsigned int *RcvDtBuf)

data	Setup data
wordNum	Number of words received
RcvDtBuf	Pointer to the buffer in which received data is stored

Description

Starts reception of serial communication and gets received data by a specified number of words. When acquisition of received data is complete, this API calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If reception of serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[ConfigSerialDriverNotify](#), [StopSerialReceiving](#)

Remark

- For the H8/300H, wait for at least a 1-bit period before calling this API after __SetSerialFormat was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- For the H8/300H, the following values are stored in the receive buffer.
High-order 8 bits: The value read from the serial status register.
Low-order 8 bits: The value read from the receive data register.
(Not read if an error occurred.)

Program example

```
#include "rapi_sif_r8c_13.h"
```

```
unsigned int buffer[10];
void func( void )
{
    .....
    /* Get 5 word data received in serial communication */
    __StartSerialReceiving( RAPI_COM1, 5, buffer );
    .....
}
```

__StartSerialSending

Synopsis

<Start transmission>

Boolean __StartSerialSending(unsigned long data, unsigned char wordNum, unsigned int *SndDtBuf)

data	Setup data
wordNum	Number of words transmitted
SndDtBuf	Pointer to the transmit data

Description

Starts transmission of serial communication and writes transmit data to the transmit buffer by a specified number of words. When transmission of all transmit data is complete, this API calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If transmission of serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__ConfigSerialDriverNotify](#), [__StopSerialSending](#)

Remark

- For the H8/300H, wait for at least a 1-bit period before calling this API after __SetSerialFormat was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"
```

```
unsigned int buffer[10];
void func( void )
{
    .....
    /* Set 5 word data to transmit buffer of serial communication */
    __StartSerialSending( RAPI_COM1, 5, buffer );
    .....
}
```


__StopSerialReceiving

Synopsis

<Stop reception>

Boolean **__StopSerialReceiving(unsigned long data)**

data	Setup data
------	------------

Description

Stops reception of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If reception of serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__StartSerialSending](#)

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    /* Stop receiving data in serial communication */
    __StopSerialReceiving ( RAPI_COM1 );
    .....
}
```

__StopSerialSending

Synopsis

<Stop transmission>

Boolean **__StopSerialSending(unsigned long data)**

data	Setup data
------	------------

Description

Stops transmission of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If transmission of serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

[__StartSerialReceiving](#)

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- When operating in clock synchronous serial communication mode, data reception is stopped at the same time by this API.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    .....
    /* Stop sending data in serial communication */
    __StopSerialSending ( RAPI_COM1 );
    .....
}
```

__PollingSerialReceiving

Synopsis

<Polling reception>

unsigned int __PollingSerialReceiving(unsigned long data)

data	Setup data
------	------------

Description

Performs reception of serial communication by polling. This API gets received data by an amount specified by __StartSerialReceiving. When acquisition of received data is complete, it calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

Out of the receive data counts requested, the number of unreceived data is returned.

Functionality

Serial I/O

Reference

[__ConfigSerialDriverNotify](#), [__SetSerialInterrupt](#), [__StartSerialReceiving](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

unsigned int buffer[10], count;

void func( void )
{
    .....
    /* Reception interrupt disable */
    __SetSerialInterrupt( RAPI_COM1 | RAPI_INT_TX_DIS | RAPI_INT_RX_DIS );
    /* Start reception */
    __StartSerialReceiving( RAPI_COM1, 5, buffer );
    do{
        count = __PollingSerialReceiving( RAPI_COM1 );
    }while(count)
    .....
}
```

__PollingSerialSending

Synopsis

<Polling transmission>

Unsigned __PollingSerialSending(unsigned long data)

data	Setup data
------	------------

Description

Performs transmission of serial communication by polling. This API sends transmit data by an amount specified by __StartSerialSending from the transmit data buffer specified by __StartSerialSending. When transmission of all transmit data is complete, it calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
-----------	-------	-----------	-------

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

Out of the transmit data counts requested, the number of untransmitted data is returned.

Functionality

Serial I/O

Reference

[__ConfigSerialDriverNotify](#), [__SetSerialInterrupt](#), [__StartSerialReceiving](#)

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"
```

```
unsigned int buffer[10],count;
void func( void )
{
    .....
    /* Transmission interrupt disable */
    __SetSerialInterrupt( RAPI_COM1 | RAPI_INT_TX_DIS | RAPI_INT_RX_DIS );
    /* Start transmission */
    __StartSerialSending( RAPI_COM1, 5, buffer );
    do{
        count = __PollingSerialSending( RAPI_COM1 );
    }while(count)
    .....
}
```

4.2.2 Timer

__CreateTimer

Synopsis

<Set timer mode>

Boolean __CreateTimer(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to timer mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_A0	Uses timer A channel 0.
RAPI_TIMER_A1	Uses timer A channel 1.
RAPI_TIMER_A2	Uses timer A channel 2.
RAPI_TIMER_A3	Uses timer A channel 3.
RAPI_TIMER_A4	Uses timer A channel 4.
RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{c32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateTimer.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateTimer.
RAPI_GATE_L	Selects a gate facility that counts a period during which input at TA_{IN} pin remains low.
RAPI_GATE_H	Selects a gate facility that counts a period during which input at TA_{IN} pin remains high.
RAPI_PULSE_ON	Selects that pulses are output from TA_{IN} pin.
RAPI_PULSE_OFF	Selects that no pulses are output from TA_{IN} pin.

• **Specifiable definition values when timer A is used (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.

- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Pulse output state) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.
- (Gate facility) Specify one from { RAPI_GATE_L, RAPI_GATE_H }. If omitted, "No gate facility" is set.

• **Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)**

- (Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(R8C)

RAPI_TIMER_X	Uses timer X.
RAPI_TIMER_Y	Uses timer Y.
RAPI_TIMER_Z	Uses timer Z.
RAPI_TIMER_RA	Uses timer RA.
RAPI_TIMER_RB	Uses timer RB.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FRING	Selects f_{RING} for the count source.
RAPI_FOCO	Selects f_{OCO} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_TIMER_Y_UNDERFLOW	Selects the underflow of timer Y for the count source.
RAPI_TIMER_RA_UNDERFLOW	Selects the underflow of timer RA for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateTimer.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateTimer.
RAPI_WRITE_RELOAD_ONLY	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to only the reload register.
RAPI_WRITE_RELOAD_BOTH	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to both the reload register and the counter, respectively.

• **Specifiable definition values when timer X is used (RAPI_TIMER_X specified)**

- (Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32 }. The default value is RAPI_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

• **Specifiable definition values when timer Y is used (RAPI_TIMER_Y specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F8, RAPI_FRING }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from { RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH }. The default value is RAPI_WRITE_RELOAD_BOTH.

• **Specifiable definition values when timer Z is used (RAPI_TIMER_Z specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_Y_UNDERFLOW}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from {RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH}. The default value is RAPI_WRITE_RELOAD_BOTH.

• **Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_FOCO, RAPI_FC32}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

• **Specifiable definition values when timer RB is used (RAPI_TIMER_RB specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_RA_UNDERFLOW}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from {RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH}. The default value is RAPI_WRITE_RELOAD_BOTH.

(H8/300H)

RAPI_TIMER_A	Uses timer A.
RAPI_TIMER_B1	Uses timer B1.
RAPI_TIMER_V	Uses timer V
RAPI_TA_F8	Supplies $\phi/8$ clock to timer A.
RAPI_TA_F32	Supplies $\phi/32$ clock to timer A.
RAPI_TA_F128	Supplies $\phi/128$ clock to timer A.
RAPI_TA_F256	Supplies $\phi/256$ clock to timer A.
RAPI_TA_F512	Supplies $\phi/512$ clock to timer A.
RAPI_TA_F2048	Supplies $\phi/2048$ clock to timer A.
RAPI_TA_F4096	Supplies $\phi/4096$ clock to timer A.
RAPI_TA_F8192	Supplies $\phi/8192$ clock to timer A.
RAPI_TB1_F4	Timer B1 counts with internal clock $\phi/4$.
RAPI_TB1_F16	Timer B1 counts with internal clock $\phi/16$.
RAPI_TB1_F64	Timer B1 counts with internal clock $\phi/64$.

RAPI_TB1_F256	Timer B1 counts with internal clock $\phi/256$.
RAPI_TB1_F512	Timer B1 counts with internal clock $\phi/512$.
RAPI_TB1_F2048	Timer B1 counts with internal clock $\phi/2048$.
RAPI_TB1_F8192	Timer B1 counts with internal clock $\phi/8192$.
RAPI_TV_F4	Timer V counts on falling edges of internal clock $\phi/4$.
RAPI_TV_F8	Timer V counts on falling edges of internal clock $\phi/8$.
RAPI_TV_F16	Timer V counts on falling edges of internal clock $\phi/16$.
RAPI_TV_F32	Timer V counts on falling edges of internal clock $\phi/32$.
RAPI_TV_F64	Timer V counts on falling edges of internal clock $\phi/64$.
RAPI_TV_F128	Timer V counts on falling edges of internal clock $\phi/128$.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateTimer.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateTimer.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_OUT_F4	Outputs $\phi/4$ clock.
RAPI_OUT_F8	Outputs $\phi/8$ clock.
RAPI_OUT_F16	Outputs $\phi/16$ clock.
RAPI_OUT_F32	Outputs $\phi/32$ clock.
RAPI_OUT_8192	Outputs $\phi W/4$ clock.
RAPI_OUT_4096	Outputs $\phi W/8$ clock.
RAPI_OUT_2048	Outputs $\phi W/16$ clock.
RAPI_OUT_1024	Outputs $\phi W/32$ clock.
RAPI_AUTO_RELOAD	Uses auto reload facility.

• **Specifiable definition values when timer A is used (RAPI_TIMER_A specified)**

- (Clock) Specify one from { RAPI_TA_F8, RAPI_TA_F32, RAPI_TA_F128, RAPI_TA_F256, RAPI_TA_F512, RAPI_TA_F2048, RAPI_TA_F4096, RAPI_TA_F8192 }. The default value is RAPI_TA_F8192.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Output) Specify one from { RAPI_OUT_F4, RAPI_OUT_F8, RAPI_OUT_F16, RAPI_OUT_F32, RAPI_OUT_8192, RAPI_OUT_4096, RAPI_OUT_2048, RAPI_OUT_1024 }. If this specification is omitted, "Clock output" is set.

• **Specifiable definition values when timer B1 is used (RAPI_TIMER_B1 specified)**

- (Clock) Specify one from { RAPI_TB1_F4, RAPI_TB1_F16, RAPI_TB1_F64, RAPI_TB1_F256, RAPI_TB1_F512, RAPI_TB1_F2048, RAPI_TB1_F8192 }. The default value is RAPI_TB1_F8192.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Reload) To select the auto reload facility, specify RAPI_AUTO_RELOAD. If RAPI_AUTO_RELOAD is not specified, "Select interval facility" is set.

• **Specifiable definition values when timer V is used (RAPI_TIMER_V specified)**

- (Clock) Specify one from { RAPI_TV_F4, RAPI_TV_F8, RAPI_TV_F16, RAPI_TV_F32, RAPI_TV_F64, RAPI_TV_F128}. If this specification is omitted, "Disable clock input" is set.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8/300H)

Specify the interrupt priority level (0-1) to be set in the interrupt control register.

For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify the value to be set in the timer register in 16 bits.

(R8C)

When using timer X, specify the set value for the timer register; when using timer Y or timer Z, specify the set value for the primary register in 8 bits.

(H8/300H)

Specify the set value for the timer reload register in 8 bits. This setting is effective only when timer B1 is used. If any timer other than B1 is used, specify 0.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C)

Specify the set value for the prescaler register in 8 bits.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (timer mode)

Reference

[_EnableTimer](#), [_DestroyTimer](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API specifies when freeing it from module standby.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}
```

```
void func( void )
{
    /* Set up timer X as timer mode */
    __CreateTimer( RAPI_TIMER_X|RAPI_TIMER_ON|RAPI_F8, 5, 0x80, 0x80,
TimerIntFunc );
}
```

EnableTimer

Synopsis

<Control operation of timer mode>

Boolean EnableTimer(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of the timer that is set to specified timer mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to timer mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to timer mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_ON	Sets the timer that is set to timer mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to timer mode to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_ON	Sets the timer that is set to timer mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to timer mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (timer mode)

Reference

[CreateTimer](#), [DestroyTimer](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Disable timer Y as timer mode */
    __EnableTimer( RAPI_TIMER_Y| RAPI_TIMER_OFF );
}
```

__DestroyTimer

Synopsis

<Discard settings of timer mode>

Boolean __DestroyTimer(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Discards settings of the timer that is set to specified timer mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (timer mode)

Reference

[__CreateTimer](#), [__EnableTimer](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"
```

```
void func( void )
{
    /* Destroy the setting of timer Z as timer mode */
    __DestroyTimer( RAPI_TIMER_Z );
}
```

__CreateEventCounter

Synopsis

<Set event counter mode>

Boolean __CreateEventCounter(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to event counter mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_A0	Uses timer A channel 0.
RAPI_TIMER_A1	Uses timer A channel 1.
RAPI_TIMER_A2	Uses timer A channel 2.
RAPI_TIMER_A3	Uses timer A channel 3.
RAPI_TIMER_A4	Uses timer A channel 4.
RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_EV_EXTERNAL	Selects the external signal input to TA _{IN} pin (when using timer Ai) or TB _{IN} pin (when using timer Bi) for the count source.
RAPI_EV_TIMER_AJ	Selects overflow or underflow of timer Aj (j = i - 1, however j = 4 if i = 0) for the count source.
RAPI_EV_TIMER_AK	Selects overflow or underflow of timer Ak (k = i - 1, however k = 0 if i = 4) for the count source.
RAPI_EV_TIMER_B2	Selects overflow or underflow of timer B2 for the count source.
RAPI_EV_TIMER_BJ	Selects overflow or underflow of timer Bj (j = i - 1, however j = 2 if i = 0) for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateEventCounter.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateEventCounter.
RAPI_PULSE_ON	Selects that pulses are output from TA _{IN} pin.
RAPI_PULSE_OFF	Selects that no pulses are output from TA _{IN} pin.
RAPI_AUTO_RELOAD	Selects reload type for the count type.
RAPI_FREE_RUN	Selects free-run type for the count type.
RAPI_UP_COUNT	Selects up-count for the count operation.
RAPI_DOWN_COUNT	Selects down-count for the count operation.

RAPI_UDF_REGISTER	Selects the UDF register for the cause of up/down switching.
RAPI_TAIOUT	Selects the input signal at TA _{OUT} pin for the cause of up/down switching.
RAPI_RISING	Selects the rising edge of count source as active edge.
RAPI_FALLING	Selects the falling edge of count source as active edge.
RAPI_BOTH	Selects both rising and falling edges of count source as active edges.

• **Specifiable definition values when timer A is used (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)**

- (Count source) Specify one from { RAPI_EV_EXTERNAL, RAPI_EV_TIMER_AJ, RAPI_EV_TIMER_AK, RAPI_EV_TIMER_B2 }. The default value is RAPI_EV_EXTERNAL.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Pulse output facility) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.
- (Gate facility) Specify one from { RAPI_GATE_L, RAPI_GATE_H }. If omitted, "No gate facility" is set.
- (Count type) Specify one from { RAPI_AUTO_RELOAD, RAPI_FREE_RUN }. The default value is RAPI_AUTO_RELOAD.
- (Count direction) Specify one from { RAPI_UP_COUNT, RAPI_DOWN_COUNT }. The default value is RAPI_DOWN_COUNT. The count direction can only be set when the UDF register is used.
- (Count direction switching) Specify one from { RAPI_UDF_REGISTER, RAPI_TAIOUT }. The default value is RAPI_UDF_REGISTER.
- (Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• **Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)**

- (Count source) Specify one from { RAPI_EV_EXTERNAL, RAPI_EV_TIMER_BJ }. The default value is RAPI_EV_EXTERNAL.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Count edge) Specify one from { RAPI_RISING, RAPI_FALLING, RAPI_BOTH }. The default value is RAPI_FALLING.

(R8C)

RAPI_TIMER_X	Uses timer X.
RAPI_TIMER_Y	Uses timer Y.
RAPI_TIMER_RA	Uses timer RA.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateEventCounter.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateEventCounter.
RAPI_WRITE_RELOAD_ONLY	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to only the reload register.

RAPI_WRITE_RELOAD_BOTH	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to both the reload register and the counter, respectively.
RAPI_RISING	Selects the rising edge of count source as active edge.
RAPI_FALLING	Selects the falling edge of count source as active edge.
RAPI_FILTER_F1	Use sampling frequency f1 for digital filter function
RAPI_FILTER_F8	Use sampling frequency f8 for digital filter function
RAPI_FILTER_F32	Use sampling frequency f32 for digital filter function
RAPI_TIOSEL_P1_7	Sets count source input pin to P1_7.
RAPI_TIOSEL_P1_5	Sets count source input pin to P1_5.
RAPI_PULSE_ON	Selects that pulses are output from TRA0 pin.
RAPI_PULSE_OFF	Selects that no pulses are output from TRA0 pin.

• **Specifiable definition values when timer X is used (RAPI_TIMER_X specified)**

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• **Specifiable definition values when timer Y is used (RAPI_TIMER_Y specified)**

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from { RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH }.
The default value is RAPI_WRITE_RELOAD_BOTH.

(INT2/CNTR1 input) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• **Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)**

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Input pin) Specify one from { RAPI_TIOSEL_P1_7, RAPI_TIOSEL_P1_5 }. The default value is RAPI_TIOSEL_P1_7.

(Pulse output function) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.

(H8/300H)

RAPI_TIMER_B1	Uses timer B1.
RAPI_TIMER_V	Uses timer V.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateEventCounter.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateEventCounter.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_AUTO_RELOAD	Uses auto reload facility.

RAPI_RISING	Selects the rising edge of count source as active edge.
RAPI_FALLING	Selects the falling edge of count source as active edge.
RAPI_BOTH	Selects both rising and falling edges of count source as active edges.
RAPI_TRGV_RISING	Selects the rising edge of TRGV pin input signal as active edge.
RAPI_TRGV_FALLING	Selects the falling edge of TRGV pin input signal as active edge.
RAPI_TRGV_BOTH	Selects both rising and falling edges of TRGV pin input signal as active edges.
RAPI_TRGV_PROHIBITED	Selects that trigger input from TRGV pin is disabled.

• **Specifiable definition values when timer B1 is used (RAPI_TIMER_B1 specified)**

- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If RAPI_OVERFLOW is not specified, "No overflow interrupt request" is set.
- (Reload) To select the auto reload facility, specify RAPI_AUTO_RELOAD. If no interrupts are specified, "No interrupt request" is set.
- (Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• **Specifiable definition values when timer V is used (RAPI_TIMER_V specified)**

- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Count edge) Specify one from { RAPI_RISING, RAPI_FALLING, RAPI_BOTH }. The default value is RAPI_FALLING.
- (TRGV pin input) Specify one from { RAPI_TRGV_RISING, RAPI_TRGV_FALLING, RAPI_TRGV_BOTH, RAPI_TRGV_PROHIBIT }. The default value is RAPI_TRGV_PROHIBIT.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8/300H)

Specify the interrupt priority level (0-1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify the value to be set in the timer register in 16 bits.

(R8C)

When using timer X, specify the set value for the timer register; when using timer Y, specify the set value for the primary register in 8 bits.

(H8/300H)

Specify the set value for the timer reload register in 8 bits. This setting is effective only when timer B1 is used. If any timer other than B1 is used, specify 0.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C)

Specify the set value for the prescaler register in 8 bits.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (event counter mode)

Reference

[__EnableEventCounter](#), [__DestroyEventCounter](#), [__GetEventCounter](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API specify when freeing it from module standby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
    /* Set up timer Y as event counter mode */
    __CreateEventCounter( RAPI_TIMER_Y|RAPI_TIMER_ON|RAPI_FALLING, 5,
                        0x80, 0x80, TimerIntFunc );
}
```

__EnableEventCounter

Synopsis

<Control operation of event counter mode>

Boolean __EnableEventCounter(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of the timer that is set to specified timer mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to event counter mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to event counter mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to event counter mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to event counter mode to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_ON	Sets the timer that is set to event counter mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to event counter mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (event counter mode)

Reference

[__CreateEventCounter](#), [__DestroyEventCounter](#), [__GetEventCounter](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c.h"

void func( void )
{
    /* Disable timer Y as event counter mode */
    __EnableEventCounter( RAPI_TIMER_Y|RAPI_TIMER_OFF );
}
```

__DestroyEventCounter

Synopsis

<Discard settings of event counter mode>

Boolean **__DestroyEventCounter(unsigned long data)**

data	Setup data (content differs with MCU type)
------	--

Description

Discards settings of the timer that is set to specified timer mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (event counter mode)

Reference

[__CreateEventCounter](#), [__EnableEventCounter](#), [__GetEventCounter](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_timer_r8c.h"
```

```
void func( void )
{
    /* Destroy the setting of timer Y as event counter mode */
    __DestroyEventCounter( RAPI_TIMER_Y );
}
```


__GetEventCounter

Synopsis

<Get event counter mode counter value>

Boolean __GetEventCounter(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which counter value is stored

Description

Gets the counter value of the timer that is set to specified event counter mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.

[data2]

Specify a pointer to the array in which the acquired counter value is stored.

(M16C)

- When using timer A (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)

[0]: The value of timer Ai register (i = 0–4) is stored.

- When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

[0]: The value of timer Bi register (i = 0–2) is stored.

(R8C)

- When using timer X (RAPI_TIMER_X specified)

[0]: The value of prescaler X register is stored.

[1]: The value of timer X register is stored.

- When using timer Y (RAPI_TIMER_Y specified)

[0]: The value of prescaler Y is stored.

[1]: The value of timer Y primary register is stored.

- When using timer RA (RAPI_TIMER_RA specified)

[0]: The value of timer RA prescaler register is stored.

[1]: The value of timer RA register is stored.

(H8/300H)

- When using timer B1 (RAPI_TIMER_B1 specified)

[0]: The value of timer counter B1 is stored.

- When using timer V (RAPI_TIMER_V specified)

[0]: The value of timer counter V is stored.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (event counter mode)

Reference

[__CreateEventCounter](#), [__EnableEventCounter](#), [__DestroyEventCounter](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    unsigned int data[2];

    /* Get the counter of timer Y as event counter mode */
    __GetEventCounter(RAPI_TIMER_Y, data );
}
```

__CreatePulseWidthModulationMode

Synopsis

<Set pulse width modulation mode>

Boolean __CreatePulseWidthModulationMode(unsigned long data1, unsigned int data2, unsigned int* data3, void* data4)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to pulse width modulation mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_A0	Uses timer A channel 0.
RAPI_TIMER_A1	Uses timer A channel 1.
RAPI_TIMER_A2	Uses timer A channel 2.
RAPI_TIMER_A3	Uses timer A channel 3.
RAPI_TIMER_A4	Uses timer A channel 4.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthModulationMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthModulationMode.
RAPI_TG_TAIIN	Selects external trigger input from TA_{iIN} pin for the count start condition.
RAPI_EV_ TIMER_AJ	Selects overflow or underflow of timer A_j ($j = i - 1$, however $j = 4$ if $i = 0$) as the trigger for the timer to start counting.
RAPI_EV_TIMER_ AK	Selects overflow or underflow of timer A_k ($k = i + 1$, however $k = 0$ if $i = 4$) as the trigger for the timer to start counting.
RAPI_EV_TIMER_ B2	Selects overflow or underflow of timer B2 as the trigger for the timer to start counting.
RAPI_TG_TAIS	Only writing 1 to the TA_S bit of the TABSR register causes the timer to start counting.
RAPI_PULSE_ON	Selects that pulses are output from TA_{iIN} pin. Selectable only when timer A_i is used.
RAPI_PULSE_OFF	Selects that no pulses are output from TA_{iIN} pin. Selectable only when timer A_i is used.
RAPI_PWM_16	Selects operation as a 16-bit pulse width modulator.
RAPI_PWM_8	Selects operation as an 8-bit pulse width modulator.

RAPI_RISING	Selects the rising edge of TA _{IN} pin input signal as active edge.
RAPI_FALLING	Selects the falling edge of TA _{IN} pin input signal as active edge.

• **Specifiable definition values when timer A is used (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)**

- (Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Count start condition) Specify one from { RAPI_TG_TAIS, RAPI_TG_TAIIN, RAPI_EV_TIMER_AJ, RAPI_EV_TIMER_AK, RAPI_EV_TIMER_B2 }. The default value is RAPI_TG_TAIIN.
- (Pulse output facility) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.
- (Modulator) Specify one from { RAPI_PWM_16, RAPI_PWM_8 }. The default value is RAPI_PWM_16.
- (TA_{IN} pin input) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING. The active edge of TA_{IN} pin input can only be set when RAPI_TG_TAIIN is selected.

(R8C)

RAPI_TIMER_Y	Uses timer Y.
RAPI_TIMER_Z	Uses timer Z.
RAPI_TIMER_RB	Uses timer RB.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_FRING	Selects f_{RING} for the count source.
RAPI_TIMER_Y_UNDERFLOW	Selects underflow of timer Y for the count source.
RAPI_TIMER_RA_UNDERFLOW	Selects underflow of timer RA for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthModulationMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthModulationMode.
RAPI_H_L_L	Sets to output a high during the primary period, a low during the secondary period, and a low when the timer is idle.
RAPI_L_H_H	Sets to output a low during the primary period, a high during the secondary period, and a high when the timer is idle.

• **Specifiable definition values when timer Y is used (RAPI_TIMER_Y specified)**

- (Count source) Specify one from { RAPI_F1, RAPI_F8, RAPI_FRING }. The default value is RAPI_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Output) Specify one from { RAPI_H_L_L, RAPI_L_H_H }. The default value is RAPI_H_L_L.

• **Specifiable definition values when timer Z is used (RAPI_TIMER_Z specified)**

- (Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_Y_UNDERFLOW}. The default value is RAPI_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Output) Specify one from { RAPI_H_L_L, RAPI_L_H_H }. The default value is RAPI_H_L_L.

• **Specifiable definition values when timer RB is used (RAPI_TIMER_RB specified)**

- (Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_RA_UNDERFLOW}. The default value is RAPI_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Output) Specify one from { RAPI_H_L_L, RAPI_L_H_H }. The default value is RAPI_H_L_L.

(H8/300H)

RAPI_TIMER_V	Uses timer V.
RAPI_TV_F4	Timer V counts on falling edges of internal clock $\phi/4$.
RAPI_TV_F8	Timer V counts on falling edges of internal clock $\phi/8$.
RAPI_TV_F16	Timer V counts on falling edges of internal clock $\phi/16$.
RAPI_TV_F32	Timer V counts on falling edges of internal clock $\phi/32$.
RAPI_TV_F64	Timer V counts on falling edges of internal clock $\phi/64$.
RAPI_TV_F128	Timer V counts on falling edges of internal clock $\phi/128$.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthModulationMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthModulationMode.
RAPI_COMPARE_MATCH_A	Enables compare match A interrupt.
RAPI_COMPARE_MATCH_B	Enables compare match B interrupt.
RAPI_OUT_H_L	Sets to output a high the during primary period and a low during the secondary period.
RAPI_OUT_L_H	Sets to output a low the during primary period and a high during the secondary period.
RAPI_RISING	Selects the rising edge of TRGV pin input signal as active edge.
RAPI_FALLING	Selects the falling edge of TRGV pin input signal as active edge.
RAPI_BOTH	Selects both rising and falling edges of TRGV pin input signal as active edge.
RAPI_TRGV_RISING	Selects the rising edge of TRGV pin input signal as active edge.
RAPI_TRGV_FALLING	Selects the falling edge of TRGV pin input signal as active edge.
RAPI_TRGV_BOTH	Selects both rising and falling edges of TRGV pin input signal as active edges.
RAPI_TRGV_PROHIBITED	Selects that trigger input from TRGV pin is disabled.

• **Specifiable definition values when timer V is used (RAPI_TIMER_V specified)**

(Clock)	Specify one from { RAPI_TV_F4, RAPI_TV_F8, RAPI_TV_F16, RAPI_TV_F32, RAPI_TV_F64, RAPI_TV_F128 }. If clock specification is omitted, specify "Disable clock input."
(Interrupt)	If compare match A interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A. If compare match B interrupt requests are enabled, specify RAPI_COMPARE_MATCH_B. If no interrupts are specified, "No interrupt request" is set.
(Operating states set)	Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
(Output)	Specify one from { RAPI_OUT_H_L, RAPI_OUT_L_H }. Always be sure to specify this output.
(TRGV pin input)	Specify one from { RAPI_TRGV_RISING, RAPI_TRGV_FALLING, RAPI_TRGV_BOTH, RAPI_TRGV_PROHIBITED }. The default value is RAPI_TRGV_PROHIBITED.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register.

For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify a pointer to the 16-bit variable in which the set value for the timer register is stored.

For 16-bit PWM, specify the value of 'n' in "high-level width n/f_j, period 65535/f_j" in 16 bits.

For 8-bit PWM, specify the values of 'n' and 'm' in "high-level width n (m + 1)/f, period 255 (m + 1)/f_j" in the 8 high-order bits and the 8 low-order bits, respectively.

(R8C)

Specify a pointer to the array in which the set value is stored.

[0]: Specify the set value for the primary register in 8 bits.

[1]: Specify the set value for the secondary register in 8 bits.

[2]: Specify the set value for the prescaler in 8 bits.

(H8/300H)

Specify a pointer to the array in which the set value is stored.

[0]: Specify the comparison value A in 8 bits.

[1]: Specify the comparison value B in 8 bits.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width modulation mode (PWM mode))

Reference

[__EnablePulseWidthModulationMode](#), [__DestroyPulseWidthModulationMode](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- For the H8/300H, make sure that comparison value A < comparison value B.
- When used for the H8/300H, this API specify when freeing it from module standby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
    unsigned int p_tim[] = {0xAA, 0xBB, 0xCC};

    /* Set up timer Z as pulse width modulation mode */
    __CreatePulseWidthModulationMode( RAPI_TIMER_Z|RAPI_TIMER_ON|RAPI_F8,
                                     5, p_tim, TimerIntFunc);
}
```

EnablePulseWidthModulationMode

Synopsis

<Control operation of pulse width modulation mode>

Boolean EnablePulseWidthModulationMode(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of the timer that is set to specified pulse width modulation mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_ON	Sets the timer that is set to pulse width modulation mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width modulation mode to stop operating.

(R8C)

RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_ON	Sets the timer that is set to pulse width modulation mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width modulation mode to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_ON	Sets the timer that is set to pulse width modulation mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width modulation mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width modulation mode (PWM mode))

Reference

[CreatePulseWidthModulationMode](#), [DestroyPulseWidthModulationMode](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer Y as pulse width modulation mode */
    __EnablePulseWidthModulationMode( RAPI_TIMER_Y|RAPI_TIMER_ON );
}
```

__DestroyPulseWidthModulationMode

Synopsis

<Discard settings of pulse width modulation mode>

Boolean **__DestroyPulseWidthModulationMode(unsigned long data)**

data	Setup data (content differs with MCU type)
------	--

Description

Discards settings of the timer that is set to specified pulse width modulation mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.

(R8C)

RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RB	Selects timer RB.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
--------------	------------------

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width modulation mode (PWM mode))

Reference

[__CreatePulseWidthModulationMode](#), [__EnablePulseWidthModulationMode](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer Z as pulse width modulation mode */
    __DestroyPulseWidthModulationMode( RAPI_TIMER_Z );
}
```

__CreatePulsePeriodMeasurementMode

Synopsis

<Set pulse period measurement mode>

Boolean __CreatePulsePeriodMeasurementMode(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to pulse period measurement mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_RISING_RISING	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_FALLING	Selects measurement of an interval from the fall to the next fall of a measurement pulse.

• **Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

(R8C)

RAPI_TIMER_X	Uses timer X.
--------------	---------------

RAPI_TIMER_RA	Uses timer RA.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_FOCO	Selects f_{OCO} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_RISING_RISING	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_FALLING	Selects measurement of an interval from the fall to the next fall of a measurement pulse.
RAPI_FILTER_F1	Use sampling frequency f_1 for digital filter function
RAPI_FILTER_F8	Use sampling frequency f_8 for digital filter function
RAPI_FILTER_F32	Use sampling frequency f_{32} for digital filter function
RAPI_TIOSEL_P1_7	Sets count source input pin to P1_7.
RAPI_TIOSEL_P1_5	Sets count source input pin to P1_5.

• **Specifiable definition values when timer X is used (RAPI_TIMER_X specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32 }. The default value is RAPI_F1.

(Operating states) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

• **Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FOCO }. The default value is RAPI_F1.

(Operating states) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measure pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Input pin) Specify one from { RAPI_TIOSEL_P1_7, RAPI_TIOSEL_P1_5 }. The default value is RAPI_TIOSEL_P1_7.

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channel 0.
RAPI_TIMER_RD1	Uses timer RD0 channel 1.
RAPI_TIMER_RD2	Uses timer RD0 channel 2.

RAPI_TIMER_RD3	Uses timer RD0 channel 3.
RAPI_TW_F1	Timer W counts with internal clock ϕ .
RAPI_TW_F2	Timer W counts with internal clock $\phi/2$.
RAPI_TW_F4	Timer W counts with internal clock $\phi/4$.
RAPI_TW_F8	Timer W counts with internal clock $\phi/8$.
RAPI_TZ_F1	Timer Z counts with internal clock ϕ .
RAPI_TZ_F2	Timer Z counts with internal clock $\phi/2$.
RAPI_TZ_F4	Timer Z counts with internal clock $\phi/4$.
RAPI_TZ_F8	Timer Z counts with internal clock $\phi/8$.
RAPI_TRC_F1	Timer RC counts with internal clock ϕ .
RAPI_TRC_F2	Timer RC counts with internal clock $\phi/2$.
RAPI_TRC_F4	Timer RC counts with internal clock $\phi/4$.
RAPI_TRC_F8	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F32	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F40M	Timer RC counts with internal clock $\phi/40M$.
RAPI_TRD_F1	Timer RD counts with internal clock ϕ .
RAPI_TRD_F2	Timer RD counts with internal clock $\phi/2$.
RAPI_TRD_F4	Timer RD counts with internal clock $\phi/4$.
RAPI_TRD_F8	Timer RD counts with internal clock $\phi/8$.
RAPI_TRD_F32	Timer RD counts with internal clock $\phi/32$.
RAPI_TRD_F40M	Timer RD counts with internal clock $\phi/40M$.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_RISING_	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RISING	
RAPI_FALLING_	Selects measurement of an interval from the fall to the next fall of a measurement pulse.
FALLING	
RAPI_TRC_FILTER_F	Use sampling frequency f1 for timer RC digital filter function
1	
RAPI_TRC_FILTER_F	Use sampling frequency f8 for timer RC digital filter function
8	
RAPI_TRC_FILTER_F	Use sampling frequency f32 for timer RC digital filter function
32	
RAPI_TRD_FILTER_F	Use sampling frequency f for timer RC digital filter function
RAPI_TRD_FILTER_F	Use sampling frequency f1 for timer RD digital filter function
1	
RAPI_TRD_FILTER_F	Use sampling frequency f8 for timer RD digital filter function
8	
RAPI_TRD_FILTER_F	Use sampling frequency f32 for timer RD digital filter function
32	
RAPI_TRD_FILTER_F	Use sampling frequency f for timer RD digital filter function
RAPI_FTIOA	Use FTIOA pin as input pin.

RAPI_FTIOB	Use FTIOB pin as input pin.
------------	-----------------------------

• **Specifiable definition values when timer W is used (RAPI_TIMER_W specified)**

- (Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.

• **Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)**

- (Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

• **Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)**

- (Count source) Specify one from { RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32 }. The default value is RAPI_RC_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.
- (Clock for digital filter) Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

• **Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)**

- (Count source) Specify one from { RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M }. The default value is RAPI_RC_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.

(Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

(Clock for digital filter) Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8/300H)

Specify the interrupt priority level (0-1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C) (H8/300H)

Specify 0.

(R8C)

When using timer X, specify the set value for the timer register.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C)

Specify the set value for the prescaler register.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse period measurement mode)

Reference

[EnablePulsePeriodMeasurementMode](#), [DestroyPulsePeriodMeasurementMode](#), [GetPulsePeriodMeasurementMode](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API specify when freeing it from module stanby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){

void func( void )
{
    /* Set up timer X as pulse period measurement mode */
    __CreatePulsePeriodMeasurementMode(
        RAPI_TIMER_X|RAPI_TIMER_ON|RAPI_FALLING_FALLING|RAPI_F8,
        5, 0x80, 0x80, TimerIntFunc);
}
```

__EnablePulsePeriodMeasurementMode

Synopsis

<Control operation of pulse period measurement mode>

Boolean __EnablePulsePeriodMeasurementMode(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of the timer that is set to specified pulse period measurement mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to pulse period measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse period measurement mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to pulse period measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse period measurement mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.
RAPI_TIMER_ON	Sets the timer that is set to pulse period measurement mode to start operating.

RAPI_TIMER_OFF	Sets the timer that is set to pulse period measurement mode to stop operating.
----------------	--

Return value If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality Timer (pulse period measurement mode)

Reference [__CreatePulsePeriodMeasurementMode](#), [__DestroyPulsePeriodMeasurementMode](#), [__GetPulsePeriodMeasurementMode](#)

- Remark**
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
 - The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer X as pulse period measurement mode */
    __EnablePulsePeriodMeasurementMode( RAPI_TIMER_X|RAPI_TIMER_ON );
}
```

__DestroyPulsePeriodMeasurementMode

Synopsis

<Discard settings of pulse period measurement mode>

Boolean __DestroyPulsePeriodMeasurementMode(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Discards settings of the timer that is set to specified pulse period measurement mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse period measurement mode)

Reference

[__CreatePulsePeriodMeasurementMode](#), [__EnablePulsePeriodMeasurementMode](#), [__GetPulsePeriodMeasurementMode](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer X as pulse period measurement mode */
    __DestroyPulsePeriodMeasurementMode( RAPI_TIMER_X );
}
```

__GetPulsePeriodMeasurementMode

Synopsis

<Get measured value in pulse period measurement mode>

Boolean __GetPulsePeriodMeasurementMode(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which counter value is stored

Description

Gets the counter value of the timer that is set to specified pulse period measurement mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.

[data2]

Specify a pointer to the array in which the acquired counter value is stored.

(M16C)

- When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

[0]: The value of timer Bi register (i = 0–2) is stored.

(R8C)

- When using timer X (RAPI_TIMER_X specified)

[0]: The value of prescaler X register is stored.

[1]: The value of timer X register is stored.

- When using timer RA (RAPI_TIMER_RA specified)

[0]: The value of timer RA prescaler register is stored.
 [1]: The value of timer RA register is stored.
 (H8/300H)

- When using timer W (RAPI_TIMER_W specified)
 [0]: (The value of general register A) – (The value of general register C) is stored.
 [1]: (The value of general register B) – (The value of general register D) is stored.
- When using timer Z (RAPI_TIMER_Z0 - RAPI_TIMER_Z1 specified)
 [0]: (The value of general register A_{i(i=0,1)}) – (The value of general register C_{i(i=0,1)}) is stored.
 [1]: (The value of general register B_{i(i=0,1)}) – (The value of general register D_{i(i=0,1)}) is stored.
- When using timer RC (RAPI_TIMER_RC specified)
 [0]: (The value of general register A) – (The value of general register C) is stored.
 [1]: (The value of general register B) – (The value of general register D) is stored.
- When using timer RD (RAPI_TIMER_RD specified)
 [0]: (The value of general register A) – (The value of general register C) is stored.
 [1]: (The value of general register B) – (The value of general register D) is stored.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (pulse period measurement mode)
Reference	CreatePulsePeriodMeasurementMode , EnablePulsePeriodMeasurementMode , DestroyPulsePeriodMeasurementMode
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.

Program example	<pre>#include "rapi_timer_r8c_13.h" void func(void) { unsigned int data[2]; /* Get the measured value of timer X as pulse period measurement mode */ __GetPulsePeriodMeasurementMode(RAPI_TIMER_X, data); }</pre>
------------------------	---

__CreatePulseWidthMeasurementMode

Synopsis

<Set pulse width measurement mode>

Boolean __CreatePulseWidthMeasurementMode(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to pulse with measurement mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthMeasurementMode.

• **Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }.
The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(R8C)

RAPI_TIMER_X	Uses timer X.
RAPI_TIMER_RA	Uses timer RA.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.

RAPI_FOCO	Selects FOCO for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_RISING_ RISING	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_ FALLING	Selects measurement of an interval from the fall to the next fall of a measurement pulse.
RAPI_FILTER_F1	Use sampling frequency f1 for digital filter function
RAPI_FILTER_F8	Use sampling frequency f8 for digital filter function
RAPI_FILTER_F32	Use sampling frequency f32 for digital filter function
RAPI_TIOSEL_P1 _7	Sets count source input pin to P1_7.
RAPI_TIOSEL_P1 _5	Sets count source input pin to P1_5.

• **Specifiable definition values when timer X is used (RAPI_TIMER_X specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32 }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_FALLING_RISING, RAPI_RISING_FALLING }. The default value is RAPI_FALLING_RISING.

• **Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FOCO }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measure pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Input pin) Specify one from { RAPI_TIOSEL_P1_7, RAPI_TIOSEL_P1_5 }. The default value is RAPI_TIOSEL_P1_7.

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channel 0.
RAPI_TIMER_RD1	Uses timer RD0 channel 1.
RAPI_TIMER_RD2	Uses timer RD0 channel 2.
RAPI_TIMER_RD3	Uses timer RD0 channel 3.
RAPI_TW_F1	Timer W counts with internal clock ϕ .
RAPI_TW_F2	Timer W counts with internal clock $\phi/2$.
RAPI_TW_F4	Timer W counts with internal clock $\phi/4$.

RAPI_TW_F8	Timer W counts with internal clock $\phi/8$.
RAPI_TZ_F1	Timer Z counts with internal clock ϕ .
RAPI_TZ_F2	Timer Z counts with internal clock $\phi/2$.
RAPI_TZ_F4	Timer Z counts with internal clock $\phi/4$.
RAPI_TZ_F8	Timer Z counts with internal clock $\phi/8$.
RAPI_TRC_F1	Timer RC counts with internal clock ϕ .
RAPI_TRC_F2	Timer RC counts with internal clock $\phi/2$.
RAPI_TRC_F4	Timer RC counts with internal clock $\phi/4$.
RAPI_TRC_F8	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F32	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F40M	Timer RC counts with internal clock $\phi/40M$.
RAPI_TRD_F1	Timer RD counts with internal clock ϕ .
RAPI_TRD_F2	Timer RD counts with internal clock $\phi/2$.
RAPI_TRD_F4	Timer RD counts with internal clock $\phi/4$.
RAPI_TRD_F8	Timer RD counts with internal clock $\phi/8$.
RAPI_TRD_F32	Timer RD counts with internal clock $\phi/32$.
RAPI_TRD_F40M	Timer RD counts with internal clock $\phi/40M$.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_RISING_	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_	Selects measurement of an interval from the fall to the next fall of a measurement pulse.
RAPI_TRC_FILTE R_F1	Use sampling frequency f1 for timer RC digital filter function
RAPI_TRC_FILTE R_F8	Use sampling frequency f8 for timer RC digital filter function
RAPI_TRC_FILTE R_F32	Use sampling frequency f32 for timer RC digital filter function
RAPI_TRD_FILTE R_F	Use sampling frequency f for timer RC digital filter function
RAPI_TRD_FILTE R_F1	Use sampling frequency f1 for timer RD digital filter function
RAPI_TRD_FILTE R_F8	Use sampling frequency f8 for timer RD digital filter function
RAPI_TRD_FILTE R_F32	Use sampling frequency f32 for timer RD digital filter function
RAPI_TRD_FILTE R_F	Use sampling frequency f for timer RD digital filter function
RAPI_FTIOA	Use FTIOA pin as input pin.
RAPI_FTIOB	Use FTIOB pin as input pin.

• Specifiable definition values when timer W is used (RAPI_TIMER_W specified)

- (Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.

• **Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)**

- (Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

• **Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)**

- (Count source) Specify one from { RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32, RAPI_TRC_F40M }. The default value is RAPI_RC_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.
- (Clock for digital filter) Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

• **Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)**

- (Count source) Specify one from { RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M }. The default value is RAPI_RC_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, "No interrupt request" is set.
- (Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.
- (Clock for digital filter) Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.
(H8/300H)

Specify the interrupt priority level (0-1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)(H8/300H)

Specify 0.

(R8C)

When using timer X, specify the set value for the timer register.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C)

Specify the set value for the prescaler register.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width measurement mode)

Reference

[__EnablePulseWidthMeasurementMode](#), [__DestroyPulseWidthMeasurementMode](#), [__GetPulseWidthMeasurementMode](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API specifies when freeing it from module standby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
    /* Set up timer X as pulse width measurement mode */
    __CreatePulseWidthMeasurementMode(
        RAPI_TIMER_X|RAPI_TIMER_ON|RAPI_RISING_FALLING|RAPI_F8,
        5, 0x80, 0x80, TimerIntFunc);
}
```

__EnablePulseWidthMeasurementMode

Synopsis

<Control operation of pulse width measurement mode>

Boolean __EnablePulseWidthMeasurementMode(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of the timer that is set to specified pulse width measurement mode.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to pulse width measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width measurement mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to pulse width measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width measurement mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.
RAPI_TIMER_ON	Sets the timer that is set to pulse width measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width measurement mode to stop operating.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (pulse width measurement mode)
Reference	CreatePulseWidthMeasurementMode , DestroyPulseWidthMeasurementMode , GetPulseWidthMeasurementMode
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Disable timer X as pulse width measurement mode */
    __EnablePulseWidthMeasurementMode( RAPI_TIMER_X|RAPI_TIMER_OFF );
}
```

__DestroyPulseWidthMeasurementMode

Synopsis

<Discard settings of pulse width measurement mode>

Boolean **__DestroyPulseWidthMeasurementMode(unsigned long data)**

data	Setup data (content differs with MCU type)
------	--

Description

Discards settings of the timer that is set to specified pulse width measurement mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
--------------	------------------

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width measurement mode)

Reference

[__CreatePulseWidthMeasurementMode](#), [__EnablePulseWidthMeasurementMode](#),
[__GetPulseWidthMeasurementMode](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer X as pulse width measurement mode */
    __DestroyPulseWidthMeasurementMode( RAPI_TIMER_X );
}
```

__GetPulseWidthMeasurementMode

Synopsis

<Get measured value in pulse width measurement mode>

Boolean __GetPulseWidthMeasurementMode(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which counter value is stored

Description

Gets the counter value of the timer that is set to specified pulse width measurement mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.

[data2]

Specify a pointer to the array in which the acquired counter value is stored.

(M16C)

- When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

[0]: The value of timer Bi register (i = 0–2) is stored.

(R8C)

- When using timer X (RAPI_TIMER_X specified)

[0]: The value of prescaler X register is stored.

[1]: The value of timer X register is stored.

- When using timer RA (RAPI_TIMER_RA specified)

[0]: The value of timer RA prescaler register is stored.
 [1]: The value of timer RA register is stored.
 (H8/300H)

- When using timer W (RAPI_TIMER_W specified)
 - [0]: (The value of general register A) – (The value of general register C) is stored.
 - [1]: (The value of general register B) – (The value of general register D) is stored.
- When using timer Z (RAPI_TIMER_Z0 - RAPI_TIMER_Z1 specified)
 - [0]: (The value of general register Ai(i=0,1)) – (The value of general register Ci(i=0,1)) is stored.
 - [1]: (The value of general register Bi(i=0,1)) – (The value of general register Di(i=0,1)) is stored.
- When using timer RC (RAPI_TIMER_RC specified)
 - [0]: (The value of general register A) – (The value of general register C) is stored.
 - [1]: (The value of general register B) – (The value of general register D) is stored.
- When using timer RD (RAPI_TIMER_RD specified)
 - [0]: (The value of general register A) – (The value of general register C) is stored.
 - [1]: (The value of general register B) – (The value of general register D) is stored.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (pulse width measurement mode)
Reference	CreatePulseWidthMeasurementMode , EnablePulseWidthMeasurementMode , DestroyPulseWidthMeasurementMode
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.

Program example	<pre>#include "rapi_timer_r8c_13.h" void func(void) { unsigned int data[2]; /* Get the measured value of timer X as pulse width measurement mode */ __GetPulseWidthMeasurementMode(RAPI_TIMER_X, data); } </pre>
------------------------	--

__CreatelInputCapture

Synopsis

<Set input capture mode>

Boolean __CreatelInputCapture(unsigned long data1, unsigned int* data2, unsigned int* data3, unsigned int* data4, void data5)**

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
data5	Setup data 5 (content differs with MCU type)

Description

Sets a specified timer to input capture mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_S	Uses timer S.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatelInputCapture.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatelInputCapture.
RAPI_OVERFLOW_BIT14	Selects overflow of bit 14 for the base timer interrupt.
RAPI_OVERFLOW_BIT15	Selects overflow of bit 15 for the base timer interrupt.

• Specifiable definition values when timer S is used (RAPI_TIMER_S specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Base timer) Specify one from { RAPI_OVERFLOW_BIT14, RAPI_OVERFLOW_BIT15 }. The default value is RAPI_OVERFLOW_BIT15.

(R8C)

RAPI_TIMER_C	Uses timer C.
RAPI_TIMER_RD0	Uses timer RD channel 0.
RAPI_TIMER_RD1	Uses timer RD channel 1.
RAPI_TRD_F1	Selects f_1 for the timer RD count source.
RAPI_TRD_F2	Selects f_2 for the timer RD count source.
RAPI_TRD_F8	Selects f_8 for the timer RD count source.
RAPI_TRD_F32	Selects f_{32} for the timer RD count source.
RAPI_TRD_F40M	Selects fOCO40M for the timer RD count source.
RAPI_FRING_FAST	Selects $f_{RING-fast}$ for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatelInputCapture.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatelInputCapture.
RAPI_RISING	Selects the rising edge of measurement pulse as active edge.

RAPI_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_F1	Uses the digital filter facility that has a sampling frequency f_1 .
RAPI_FILTER_F8	Uses the digital filter facility that has a sampling frequency f_8 .
RAPI_FILTER_F32	Uses the digital filter facility that has a sampling frequency f_{32} .
RAPI_INT3_TRIGGER	Selects the trigger facility of TC _{IN} input.
RAPI_FRING128	Selects the trigger facility of f _{RING128} input.
RAPI_OVERFLOW	Enable overflow interrupt.
RAPI_INPUT_CAPTU RE_A	Enable GRA input capture interrupt.
RAPI_INPUT_CAPTU RE_B	Enable GRB input capture interrupt.
RAPI_INPUT_CAPTU RE_C	Enable GRC input capture interrupt.
RAPI_INPUT_CAPTU RE_D	Enable GRD input capture interrupt.
RAPI_COUNT_CLEA R_A	Selects GRA input capture to counter clear factor
RAPI_COUNT_CLEA R_B	Selects GRB input capture to counter clear factor
RAPI_COUNT_CLEA R_C	Selects GRC input capture to counter clear factor
RAPI_COUNT_CLEA R_D	Selects GRD input capture to counter clear factor
RAPI_COUNT_CLEA R_SYNC	Clear counter in sync with the synchronized other timer counter
RAPI_TIMER_SYNC	Synchronize timer on channels A and B.
RAPI_TRD_FILTER_F 1	Use sampling frequency f_1 for digital filter function
RAPI_TRD_FILTER_F 8	Use sampling frequency f_8 for digital filter function
RAPI_TRD_FILTER_F 32	Use sampling frequency f_{32} for digital filter function
RAPI_TRD_FILTER_F	Use sampling frequency f for digital filter function

• **Specifiable definition values when timer C is used (RAPI_TIMER_C specified)**

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF

- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. When not specifying interrupts, select "No interrupt requests."
- (Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.
- (Synchronization) If timers on channels 0 and 1 are to be synchronized, select RAPI_TIMER_SYNC. When not specifying synchronization, select "Channels 0 and 1 operate independently."
- (Clock for digital filter) Specify one from [RAPI_TRD_FILTER_F1, RAPI_TRD_FILTER_F8, RAPI_TRD_FILTER_F32, RAPI_TRD_FILTER_F]. The default is RAPI_TRD_FILTER_F32.

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channel 0.
RAPI_TIMER_RD1	Uses timer RD0 channel 1.
RAPI_TIMER_RD2	Uses timer RD0 channel 2.
RAPI_TIMER_RD3	Uses timer RD0 channel 3.
RAPI_TW_F1	Timer W counts with internal clock ϕ .
RAPI_TW_F2	Timer W counts with internal clock $\phi/2$.
RAPI_TW_F4	Timer W counts with internal clock $\phi/4$.
RAPI_TW_F8	Timer W counts with internal clock $\phi/8$.
RAPI_TZ_F1	Timer Z counts with internal clock ϕ .
RAPI_TZ_F2	Timer Z counts with internal clock $\phi/2$.
RAPI_TZ_F4	Timer Z counts with internal clock $\phi/4$.
RAPI_TZ_F8	Timer Z counts with internal clock $\phi/8$.
RAPI_TRC_F1	Timer RC counts with internal clock ϕ .
RAPI_TRC_F2	Timer RC counts with internal clock $\phi/2$.
RAPI_TRC_F4	Timer RC counts with internal clock $\phi/4$.
RAPI_TRC_F8	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F32	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F40M	Timer RC counts with internal clock $\phi/40M$.
RAPI_TRD_F1	Timer RD counts with internal clock ϕ .
RAPI_TRD_F2	Timer RD counts with internal clock $\phi/2$.

RAPI_TRD_F4	Timer RD counts with internal clock $\phi/4$.
RAPI_TRD_F8	Timer RD counts with internal clock $\phi/8$.
RAPI_TRD_F32	Timer RD counts with internal clock $\phi/32$.
RAPI_TRD_F40M	Timer RD counts with internal clock $\phi/40M$.
RAPI_TIMER_ON	Sets the timer to start operating in <code>__CreateInputCapture</code> .
RAPI_TIMER_OFF	Sets the timer to stop operating in <code>__CreateInputCapture</code> .
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_INPUT_CAPTURE_A	Enables GRA input capture interrupt.
RAPI_INPUT_CAPTURE_B	Enables GRB input capture interrupt.
RAPI_INPUT_CAPTURE_C	Enables GRC input capture interrupt.
RAPI_INPUT_CAPTURE_D	Enables GRD input capture interrupt.
RAPI_COUNT_CLEAR_A	Specifies GRA input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_B	Specifies GRB input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_C	Specifies GRC input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_D	Specifies GRD input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_SYNC	Clear counter in sync with the synchronized other timer counter
RAPI_TIMER_SYNC	Synchronizes timer on channels 0 and 1.
RAPI_TRC_FILTER_F1	Use sampling frequency f1 for timer RC digital filter function
RAPI_TRC_FILTER_F8	Use sampling frequency f8 for timer RC digital filter function
RAPI_TRC_FILTER_F32	Use sampling frequency f32 for timer RC digital filter function
RAPI_TRC_FILTER_F	Use sampling frequency f for timer RC digital filter function
RAPI_TRD_FILTER_F1	Use sampling frequency f1 for timer RD digital filter function
RAPI_TRD_FILTER_F8	Use sampling frequency f8 for timer RD digital filter function
RAPI_TRD_FILTER_F32	Use sampling frequency f32 for timer RD digital filter function
RAPI_TRD_FILTER_F	Use sampling frequency f for timer RD digital filter function

• **Specifiable definition values when timer W is used (RAPI_TIMER_W specified)**

- (Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. If no interrupts are specified, "No interrupt request" is set.

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.

• **Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)**

(Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. If no interrupts are specified, "No interrupt request" is set.

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause of counter clear, specify RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively.

(Synchronization) If the timer is synchronized on channels 0 and 1, specify RAPI_TIMER_SYNC. If synchronization is not specified, "Channels 0 and 1 operate independently" is set.

• **Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)**

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. When not specifying interrupts, select "No interrupt requests."

(Clock for digital filter) Specify one from [RAPI_TRD_FILTER_F1, RAPI_TRD_FILTER_F8, RAPI_TRD_FILTER_F32, RAPI_TRD_FILTER_F]. The default is RAPI_TRD_FILTER_F32.

• **Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)**

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set)	Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.
(Interrupt)	If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. When not specifying interrupts, select "No interrupt requests."
(Counter clear)	To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.
(Synchronization)	If timers on channels 0 and 1 are to be synchronized, select RAPI_TIMER_SYNC. When not specifying synchronization, select "Channels 0 and 1 operate independently."
(Clock for digital filter)	Specify one from [RAPI_TRD_FILTER_F1, RAPI_TRD_FILTER_F8, RAPI_TRD_FILTER_F32, RAPI_TRD_FILTER_F]. The default is RAPI_TRD_FILTER_F32.

[data2]

(M16C)

Specify a pointer to the array in which the interrupt priority level is stored.

[0]: Specify the IC/OC base timer interrupt priority level (0-7).

[1]: Specify the IC/OC interrupt 0 priority level (0-7).

[2]: Specify the IC/OC interrupt 1 priority level (0-7).

(R8C)

When timer RD is used (RAPI_TIMER_RD0 or RAPI_TIMER_RD1 specified), specify a pointer to the variable that contains the interrupt priority level (0-7) to be set in the interrupt control register. When timer RD is not used, specify a pointer to the following array that contains the interrupt priority level

[0]: Specify the timer C interrupt priority level (0-7).

[1]: Specify the compare match 0 interrupt priority level (0-7).

[2]: Specify the compare match 1 interrupt priority level (0-7).

(H8/300H)

Specify the interrupt priority level (0-1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify a pointer to the array in which the set value for the time measurement control register is stored.

[0]: Specify the set value for time measurement control register 0.

[1]: Specify the set value for time measurement control register 1.

- [2]: Specify the set value for time measurement control register 2.
 [3]: Specify the set value for time measurement control register 3.
 [4]: Specify the set value for time measurement control register 4.
 [5]: Specify the set value for time measurement control register 5.
 [6]: Specify the set value for time measurement control register 6.
 [7]: Specify the set value for time measurement control register 7.

For each element of the array, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

RAPI_IC_RISING	Selects the rising edge of measurement pulse as active edge.
RAPI_IC_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_IC_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_F1_F2	Uses the digital filter facility that has a sampling frequency f_1 or f_2 .
RAPI_FILTER_FBT1	Uses the digital filter facility that has a sampling frequency f_{BT1} .
RAPI_GATE	Uses a gate facility.
RAPI_GATE_CLEAR	Clears the gate facility upon a match of base timer and G1POK register.
RAPI_PRESCALER	Uses a prescaler facility.

• **Specifiable definition values for time measurement registers 0-7**

(Measurement pulse) Specify one from { RAPI_IC_RISING, RAPI_IC_FALLING, RAPI_IC_BOTH }. If no measurement pulses are specified, the time measurement register is set to “No time measurement performed.”

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_FBT1 }. If no filters are specified, “No digital filter” is set.

• **Specifiable definition values for time measurement registers 6-7**

(Gate) To use the gate facility, specify RAPI_GATE. If RAPI_GATE is not specified, “No facility unused” is set. Make sure RAPI_GATE_CLEAR is specified at the same time RAPI_GATE is specified.

(Prescaler) To use the prescaler facility, specify RAPI_PRESCALER. If RAPI_PRESCALER is not specified, “No prescaler facility” is set.

(R8C)

When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD1 specified), set one of the following values. Otherwise, specify 0.

Specify a pointer to the array in which each active edge setting is stored.

[0]: Specify the active edge of TRDIOAi (i=0,1) pin.

[1]: Specify the active edge of TRDIOBi (i=0,1) pin.

[2]: Specify the active edge of TRDIOCi (i=0,1) pin.

[3]: Specify the active edge of TRDIODi (i=0,1) pin.

For each element of the array, one of [RAPI_TRD_RISING, RAPI_TRD_FALLING, RAPI_TRD_BOTH] can be set as the active edge of measured pulses.

Furthermore, if the digital filter function is enabled, select RAPI_FILTER_ON.

To set the f0C0128 signal for the TRDIOA0 pin on channel 0, specify RAPI_F0C0128. For the elements corresponding to unused channels, set 0.

RAPI_TRD_RISING	Selects the rising edge of measurement pulse as active edge.
RAPI_TRD_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_TRD_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_ON	Selects digital filter on. If not select "RAPI_FILTER_ON", "No digital filter" is set.
RAPI_FOCO128	Selects fOCO128 as TRDIOA0 pin input. Specifiable only active edge of channel0 TRDIOA0 pin.

(H8/300H)

Specify a pointer to the array in which each active edge setting is stored.

[0]: Specify the active edge of TRDIOA pin.

[1]: Specify the active edge of TRDIOB pin.

[2]: Specify the active edge of TRDIOC pin.

[3]: Specify the active edge of TRDIOD pin.

For each element of the array, one of the following definition values { RAPI_RISING, RAPI_FALLING, RAPI_BOTH } can be set.

For the array elements corresponding to unused channels, set 0.

RAPI_RISING	Selects the rising edge of measurement pulse as active edge.
RAPI_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_ON	Selects digital filter on. If not select "RAPI_FILTER_ON", "No digital filter" is set.

[data4]

(M16C)

Specify a pointer to the array in which the set value for each register or timer S is stored.

[0]: Specify the set value for the facility select and facility enable registers.

Specify the channel for which the time measurement facility is enabled.

[1]: Specify the set value for interrupt enable register 0.

Specify the channel for which IO/CO interrupt 0 request is enabled.

[2]: Specify the set value for interrupt enable register 1.

Specify the channel for which IO/CO interrupt 1 request is enabled.

[3]: Specify the set value for the count source divide-by-N register.

Specify the value of 'n' in the formula "count source divided by (n + 1)" in 8 bits.

[4]: Specify the set value for time measurement prescaler register 6.

Specify the value of 'n' in the prescaler period "n + 1" in 8 bits.

[5]: Specify the set value for time measurement prescaler register 7.

Specify the value of 'n' in the prescaler period "n + 1" in 8 bits.

For the channels to be specified in each array element, use the following definition values. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. If 0 is specified, the value 0 is set in the corresponding register.

RAPI_CHANNEL0	Selects channel 0.
---------------	--------------------

RAPI_CHANNEL1	Selects channel 1.
RAPI_CHANNEL2	Selects channel 2.
RAPI_CHANNEL3	Selects channel 3.
RAPI_CHANNEL4	Selects channel 4.
RAPI_CHANNEL5	Selects channel 5.
RAPI_CHANNEL6	Selects channel 6.
RAPI_CHANNEL7	Selects channel 7.

(R8C) (H8/300H)

Specify 0.

[data5]

(M16C)

Specify a pointer to the array in which the callback function is stored.

[0]: Specify a pointer to the callback function for IC/OC base timer interrupt. If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for IC/OC interrupt 0. If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for IC/OC interrupt 1. If this pointer is not specified, 0 is set.

(R8C) (H8/300H)

Specify a pointer to the array in which the callback function is stored. If this pointer is not specified, RAPI_NULL is set.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (input capture mode)

Reference

[__EnableInputCapture](#), [__DestroyInputCapture](#), [__GetInputCapture](#)

Remark

- If an undefined value is specified in the first, third and fourth arguments, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When use CPU which has no digital filter function, cannot specify digital filter setting.
- When used for the H8/300H, this API specify when freeing it from module stanby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}
```

```
void func( void )
{
    /* Set up timer C as input capture mode */
    __CreateInputCapture(
        RAPI_TIMER_C|RAPI_TIMER_ON|RAPI_BOTH|RAPI_F32| RAPI_FRING128,
        5, TimerIntFunc);
}
```

__EnableInputCapture

Synopsis

<Control operation of input capture mode>

Boolean **__EnableInputCapture**(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of the timer that is set to specified input capture mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_S	Selects timer S.
RAPI_TIMER_ON	Sets the timer that is set to input capture mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to input capture mode to stop operating.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RD0	Selects timer RD0 channel 0.
RAPI_TIMER_RD1	Selects timer RD0 channel 1.
RAPI_TIMER_ON	Sets the timer that is set to input capture mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to input capture mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channel 0.
RAPI_TIMER_RD1	Selects timer RD0 channel 1.
RAPI_TIMER_RD2	Selects timer RD0 channel 2.
RAPI_TIMER_RD3	Selects timer RD0 channel 3.
RAPI_TIMER_ON	Sets the timer that is set to input capture mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to input capture mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (input capture mode)

Reference

[CreateInputCapture](#), [DestroyInputCapture](#), [GetInputCapture](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer C as input capture mode */
    __EnableInputCapture( RAPI_TIMER_C|RAPI_TIMER_ON );
}
```

__DestroyInputCapture

Synopsis

<Discard settings of input capture mode>

Boolean **__DestroyInputCapture(unsigned long data)**

data	Setup data (content differs with MCU type)
------	--

Description

Discards settings of the timer that is set to specified input capture mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_S	Selects timer S.
--------------	------------------

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (input capture mode)

Reference

[__CreateInputCapture](#), [__EnableInputCapture](#), [__GetInputCapture](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"
```

```
void func( void )
{
    /* Destroy the setting of timer C as input capture mode */
    __DestroyInputCapture( RAPI_TIMER_C );
}
```

__GetInputCapture

Synopsis

<Get input capture mode counter value>

Boolean __GetInputCapture(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)

Description

Gets the counter value of the timer that is set to specified input capture mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_S	Selects timer S.
--------------	------------------

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.

[data2]

(M16C)

Specify a pointer to the array in which the acquired counter value is stored.

[0]: Stores the value of base timer register 0.

[1]: Stores the value of time measurement register 0.

[2]: Stores the value of time measurement register 1.

[3]: Stores the value of time measurement register 2.

[4]: Stores the value of time measurement register 3.

[5]: Stores the value of time measurement register 4.

[6]: Stores the value of time measurement register 5.

[7]: Stores the value of time measurement register 6.

[8]: Stores the value of time measurement register 7.

(R8C)

Specify a pointer to the array in which the acquired counter value is stored.

• **When timer C is used (RAPI_TIMER_C specified)**

[0]: Stores the value of timer C counter.

[1]: Stores the value of capture & compare 0 register.

- When timer RD is used (RAPI_TIMER_RD0- RAPI_TIMER_RD1 specified)

[0]: Stores the value of timer counter.
 [1]: Stores the value of general register A.
 [2]: Stores the value of general register B.
 [3]: Stores the value of general register C.
 [4]: Stores the value of general register D.

(H8/300H)

[0]: Stores the value of the timer counter.
 [1]: Stores the value of general register A.
 [2]: Stores the value of general register B.
 [3]: Stores the value of general register C.
 [4]: Stores the value of general register D.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (input capture mode)
Reference	__CreateInputCapture , __EnableInputCapture , __DestroyInputCapture
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.
Program example	

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    unsigned int data[2];

    /* Get the counter of timer C as input capture mode */
    __GetInputCapture( RAPI_TIMER_C, data );
}
```


__CreateOutputCompare

Synopsis

<Set output compare mode>

Boolean __CreateOutputCompare(unsigned long data1, unsigned int* data2, unsigned int* data3, unsigned int* data4, void data5)**

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
data5	Setup data 5 (content differs with MCU type)

Description

Sets a specified timer to output compare mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_S	Uses timer S.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateOutputCompare.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateOutputCompare.
RAPI_OVERFLOW_BIT14	Selects overflow of bit 14 for the base timer interrupt.
RAPI_OVERFLOW_BIT15	Selects overflow of bit 15 for the base timer interrupt.

• Specifiable definition values when timer S is used (RAPI_TIMER_S specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Base timer) Specify one from { RAPI_OVERFLOW_BIT14, RAPI_OVERFLOW_BIT15 }. The default value is RAPI_OVERFLOW_BIT15.

(R8C)

RAPI_TIMER_C	Uses timer C.
RAPI_TIMER_RD0	Uses timer RD0 channe 0.
RAPI_TIMER_RD1	Uses timer RD0 channe 1.
RAPI_TIMER_RE	Uses timer C.
RAPI_F1	Selects f_1 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FRING_FAST	Selects $f_{RING-fast}$ for the count source.
RAPI_TRD_F1	Selects f_1 for the timer RD count source.
RAPI_TRD_F2	Selects f_2 for the timer RD count source.
RAPI_TRD_F4	Selects f_4 for the timer RD count source.
RAPI_TRD_F8	Selects f_8 for the timer RD count source.

RAPI_TRD_F32	Selects f_{32} for the timer RD count source.
RAPI_TRD_F40M	Selects fOCO40M for the timer RD count source.
RAPI_TRE_F4	Selects f_4 for the timer RE count source.
RAPI_TRE_F8	Selects f_8 for the timer RE count source.
RAPI_TRE_F32	Selects f_{32} for the timer RE count source.
RAPI_TRE_FC4	Selects fC4 for the timer RE count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateOutputCompare.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateOutputCompare.
RAPI_TIMER_OVERFLOW	Enables the overflow interrupt.
RAPI_COMPARE_MATCH	Enables the comparematch interrupt.
RAPI_COMPARE_MATCH_A	Enables the GRA compare match interrupt.
RAPI_COMPARE_MATCH_B	Enables the GRB compare match interrupt.
RAPI_COMPARE_MATCH_C	Enables the GRC compare match interrupt.
RAPI_COMPARE_MATCH_D	Enables the GRD compare match interrupt.
RAPI_COUNT_CLEAR_A	Specifies GRA compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_B	Specifies GRB compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_C	Specifies GRC compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_D	Specifies GRD compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_SYNC	Clear counter in sync with the synchronized other timer counter
RAPI_TIMER_SYNC	Synchronize timer on channels 0 and 1.
RAPI_CMP00_DISABLE	Disables CMP output from CMP0 ₀ .
RAPI_CMP01_DISABLE	Disables CMP output from CMP0 ₁ .
RAPI_CMP02_DISABLE	Disables CMP output from CMP0 ₂ .
RAPI_CMP10_DISABLE	Disables CMP output from CMP1 ₀ .
RAPI_CMP11_DISABLE	Disables CMP output from CMP1 ₁ .
RAPI_CMP12_DISABLE	Disables CMP output from CMP1 ₂ .
RAPI_CMP00_ENABLE	Enables CMP output from CMP0 ₀ .
RAPI_CMP01_ENABLE	Enables CMP output from CMP0 ₁ .
RAPI_CMP02_ENABLE	Enables CMP output from CMP0 ₂ .
RAPI_CMP10_ENABLE	Enables CMP output from CMP1 ₀ .
RAPI_CMP11_ENABLE	Enables CMP output from CMP1 ₁ .
RAPI_CMP12_ENABLE	Enables CMP output from CMP1 ₂ .
RAPI_OUTPUT_REVERSE_0	Inverts CMP output from CMP0 ₀ through CMP0 ₂ .
RAPI_OUTPUT_REVERSE_1	Inverts CMP output from CMP1 ₀ through CMP1 ₂ .
RAPI_RELOAD	Sets TC register to "0x0000" when compare 1 matches.

RAPI_UNCHANGE_0	Leaves CMP0 output unchanged even when matched in compare 0.
RAPI_REVERSE_0	Inverts CMP0 output upon compare 0 match signal.
RAPI_L_0	Forces CMP0 output low upon compare 0 match signal.
RAPI_H_0	Forces CMP0 output high upon compare 0 match signal.
RAPI_UNCHANGE_1	Leaves CMP1 output unchanged even when matched in compare 1.
RAPI_REVERSE_1	Inverts CMP1 output upon compare 1 match signal.
RAPI_L_1	Forces CMP1 output low upon compare 1 match signal.
RAPI_H_1	Forces CMP1 output high upon compare 1 match signal.
RAPI_OUTPUT_DISABLE	Disable output.
RAPI_OUTPUT_F2	Specifies f2 output for output function.
RAPI_OUTPUT_F4	Specifies f4 output for output function.
RAPI_OUTPUT_F8	Specifies f8 output for output function.
RAPI_OUTPUT_COMPARE	Specifies compare output for output function.
RAPI_4BIT_COUNTER	Uses 4 bit counter.
RAPI_COMPARE_MATCH_A_STOP	Stop count when GRA compare match occur
RAPI_STOP	Stop count when clear TSTART bit.

• **Specifiable definition values when timer C is used (RAPI_TIMER_C specified)**

(Count source)	Specify one from { RAPI_F1, RAPI_F8, RAPI_F32, RAPI_FRING_FAST }. The default value is RAPI_F1.
(Operating states set)	Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
(CMP0 ₀ output)	Specify one from { RAPI_CMP00_DISABLE, RAPI_CMP00_ENABLE }. The default value is RAPI_CMP00_DISABLE.
(CMP0 ₁ output)	Specify one from { RAPI_CMP01_DISABLE, RAPI_CMP01_ENABLE }. The default value is RAPI_CMP01_DISABLE.
(CMP0 ₂ output)	Specify one from { RAPI_CMP02_DISABLE, RAPI_CMP02_ENABLE }. The default value is RAPI_CMP02_DISABLE.
(CMP1 ₀ output)	Specify one from { RAPI_CMP10_DISABLE, RAPI_CMP10_ENABLE }. The default value is RAPI_CMP10_DISABLE.
(CMP1 ₁ output)	Specify one from { RAPI_CMP11_DISABLE, RAPI_CMP11_ENABLE }. The default value is RAPI_CMP11_DISABLE.
(CMP1 ₂ output)	Specify one from { RAPI_CMP12_DISABLE, RAPI_CMP12_ENABLE }. The default value is RAPI_CMP12_DISABLE.
(CMP0 output inversion)	To invert CMP0 output, specify RAPI_OUTPUT_REVERSE_0. If RAPI_OUTPUT_REVERSE_0 is not specified, "CMP0 output not inverted" is set.
(CMP1 output inversion)	To invert CMP1 output, specify RAPI_OUTPUT_REVERSE_1. If RAPI_OUTPUT_REVERSE_1 is not specified, "CMP1 output not inverted" is set.

- (TC reload) To reload TC register, specify RAPID_RELOAD. If RAPID_RELOAD is not specified, "No reload" is set.
- (CMP0 output mode) Specify one from { RAPI_UNCHANGE_0, RAPI_REVERSE_0, RAPI_L_0, RAPI_H_0 }. The default value is RAPI_UNCHANGE_0.
- (CMP1 output mode) Specify one from { RAPI_UNCHANGE_1, RAPI_REVERSE_1, RAPI_L_1, RAPI_H_1 }. The default value is RAPI_UNCHANGE_1.

• **Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)**

- (Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.
- (Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. When not specifying interrupts, select "No interrupt requests."
- (Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.
- (Synchronization) If timers on channels 0 and 1 are to be synchronized, select RAPI_TIMER_SYNC. When not specifying synchronization, select "Channels 0 and 1 operate independently."
- (count stop) Specify one from [RAPI_COMPARE_MATCH_A_STOP, RAPI_STOP]. The default is RAPI_STOP.

• **Specifiable definition values when timer RE is used (RAPI_TIMER_RE specified)**

- (Count source) Specify one from [RAPI_TRE_F4, RAPI_TRE_F8, RAPI_TRE_F32, RAPI_TRD_FC4]. The default is RAPI_TRD_F1.
- (Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF
- (Interrupt) If compare match interrupt requests are enabled, specify RAPI_COMPARE_MATCH. When not specifying interrupts, select "No interrupt requests."
- (Output function) Specify one from [RAPI_OUTPUT_DISABLE, RAPI_OUTPUT_F2, RAPI_OUTPUT_F4, RAPI_OUTPUT_F8, RAPI_OUTPUT_COMPARE]. The default is RAPI_OUTPUT_COMPARE.
- (Counter) If use 4 bit counter, specify RAPI_4BIT_COUNTER. The default is 8 bit counter .

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channel 0.
RAPI_TIMER_RD1	Uses timer RD0 channel 1.
RAPI_TIMER_RD2	Uses timer RD0 channel 2.
RAPI_TIMER_RD3	Uses timer RD0 channel 3.
RAPI_TW_F1	Timer W counts with internal clock ϕ .
RAPI_TW_F2	Timer W counts with internal clock $\phi/2$.
RAPI_TW_F4	Timer W counts with internal clock $\phi/4$.
RAPI_TW_F8	Timer W counts with internal clock $\phi/8$.
RAPI_TZ_F1	Timer Z counts with internal clock ϕ .
RAPI_TZ_F2	Timer Z counts with internal clock $\phi/2$.
RAPI_TZ_F4	Timer Z counts with internal clock $\phi/4$.
RAPI_TZ_F8	Timer Z counts with internal clock $\phi/8$.
RAPI_TRC_F1	Timer RC counts with internal clock ϕ .
RAPI_TRC_F2	Timer RC counts with internal clock $\phi/2$.
RAPI_TRC_F4	Timer RC counts with internal clock $\phi/4$.
RAPI_TRC_F8	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F32	Timer RC counts with internal clock $\phi/8$.
RAPI_TRC_F40M	Timer RC counts with internal clock $\phi/40M$.
RAPI_TRD_F1	Timer RD counts with internal clock ϕ .
RAPI_TRD_F2	Timer RD counts with internal clock $\phi/2$.
RAPI_TRD_F4	Timer RD counts with internal clock $\phi/4$.
RAPI_TRD_F8	Timer RD counts with internal clock $\phi/8$.
RAPI_TRD_F32	Timer RD counts with internal clock $\phi/32$.
RAPI_TRD_F40M	Timer RD counts with internal clock $\phi/40M$.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateOutputCompare.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateOutputCompare.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_COMPARE_ MATCH_A	Enables GRA compare match interrupt.
RAPI_COMPARE_ MATCH_B	Enables GRB compare match interrupt.
RAPI_COMPARE_ MATCH_C	Enables GRC compare match interrupt.
RAPI_COMPARE_ MATCH_D	Enables GRD compare match interrupt.
RAPI_COUNT_CLEAR_A	Specifies GRA compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_B	Specifies GRB compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_C	Specifies GRC compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_D	Specifies GRD compare match for the cause of counter clear.

RAPI_COUNT_CLEAR_S YNC	Clear counter in sync with the synchronized other timer counter
RAPI_TIMER_SYNC	Synchronizes timer on channels 0 and 1.

• **Specifiable definition values when timer W is used (RAPI_TIMER_W specified)**

- (Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_ATCH_A, RAPI_COMPARE_ATCH_B, RAPI_COMPARE_ATCH_C, or RAPI_COMPARE_ATCH_D, respectively. If no interrupts are specified, "No interrupt request" is set.
- (Counter clear) To specify GRA compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A.

• **Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)**

- (Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.
- (Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
- (Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_ATCH_A, RAPI_COMPARE_ATCH_B, RAPI_COMPARE_ATCH_C, or RAPI_COMPARE_ATCH_D, respectively. If no interrupts are specified, "No interrupt request" is set.
- (Counter clear) To specify GRA, GRB, GRC, or GRD compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively.
- (Synchronization) If the timer is synchronized on channels 0 and 1, specify RAPI_TIMER_SYNC. If synchronization is not specified, "Channels 0 and 1 operate independently" is set.

• **Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)**

- (Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.
- (Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A, RAPI_COMPARE_MATCH_B, RAPI_COMPARE_MATCH_C, or RAPI_COMPARE_MATCH_D, respectively. When not specifying interrupts, select "No interrupt requests."

(Counter clear) To specify GRA compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A.

• **Specifiable definition values when timer RD is used (RAPI_TIMER_RD0 to RAPI_TIMER_RD4 specified)**

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A, RAPI_COMPARE_MATCH_B, RAPI_COMPARE_MATCH_C, or RAPI_COMPARE_MATCH_D, respectively. If no interrupts are specified, "No interrupt request" is set.

(Counter clear) To specify GRA, GRB, GRC, or GRD compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively.

(Synchronization) If the timer is synchronized on channels 0 and 1, specify RAPI_TIMER_SYNC. If synchronization is not specified, "Channels 0 and 1 operate independently" is set.

[data2]

(M16C)

Specify a pointer to the array in which the interrupt priority level is stored.

[0]: Specify the IC/OC base timer interrupt priority level (0-7).

[1]: Specify the IC/OC interrupt 0 priority level (0-7).

[2]: Specify the IC/OC interrupt 1 priority level (0-7).

(R8C)

Specify a pointer to the array in which the interrupt priority level is stored.

[0]: Specify the timer C interrupt priority level (0-7).

[1]: Specify the compare match 0 interrupt priority level (0-7).

[2]: Specify the compare match 1 interrupt priority level (0-7).

(H8/300H)

Specify 0.

[data3]

(M16C)

Specify a pointer to the array in which the set value for the waveform generation control register or waveform generation register is stored.

[0]: Specify the set value for waveform generation control register 0.

[1]: Specify the set value for waveform generation control register 1.

[2]: Specify the set value for waveform generation control register 2.

[3]: Specify the set value for waveform generation control register 3.

[4]: Specify the set value for waveform generation control register 4.

[5]: Specify the set value for waveform generation control register 5.

[6]: Specify the set value for waveform generation control register 6.

[7]: Specify the set value for waveform generation control register 7.

[8]: Specify the set value for waveform generation register 0 in 16 bits.

Specify the comparison value of channel 0 in 16 bits.

[9]: Specify the set value for waveform generation register 1 in 16 bits.

Specify the comparison value of channel 1 in 16 bits.

[10]: Specify the set value for waveform generation register 2 in 16 bits.

Specify the comparison value of channel 2 in 16 bits.

[11]: Specify the set value for waveform generation register 3 in 16 bits.

Specify the comparison value of channel 3 in 16 bits.

[12]: Specify the set value for waveform generation register 4 in 16 bits.

Specify the comparison value of channel 4 in 16 bits.

[13]: Specify the set value for waveform generation register 5 in 16 bits.

Specify the comparison value of channel 5 in 16 bits.

[14]: Specify the set value for waveform generation register 6 in 16 bits.

Specify the comparison value of channel 6 in 16 bits.

[15]: Specify the set value for waveform generation register 7 in 16 bits.

Specify the comparison value of channel 7 in 16 bits.

To specify the set value for each waveform generation control register that is an array element, the following definition values can be set. To specify multiple definition values at the same time, use the symbol “|” to separate each specified value.

RAPI_SINGLE	Selects single-phase waveform output mode.
RAPI_SR	Selects SR waveform output mode.
RAPI_PHASE_DELAYED	Selects inverted waveform output mode.
RAPI_OUT_INIT_0	Outputs a 0 as the initial output value.
RAPI_OUT_INIT_1	Outputs a 1 as the initial output value.
RAPI_OUTPUT_REVERSED	Uses an output inversion facility.
RAPI_RELOAD_WITH_RESET	Selects a reset of the base timer as the timing with which the G1P0J register value is reloaded.
RAPI_RELOAD_WITH_WRITE	Selects a write to the base timer as the timing with which the G1P0J register value is reloaded.

• Specifiable definition values for waveform generation control registers 0-7

- (Output mode) Specify one from { RAPI_SINGLE, RAPI_SR, RAPI_PHASE_DELAYED }.
The default value is RAPI_SINGLE.
- (Initial output value) Specify one from { RAPI_OUT_INIT_0, RAPI_OUT_INIT_1 }. The default value is RAPI_OUT_INIT_0.
- (Output inversion) To invert the output, specify RAPI_OUTPUT_REVERSED. If output inversion is not specified, "Output not inverted" is set.
- (Reload) Specify one from { RAPI_RELOAD_WITH_RESET, RAPI_RELOAD_WITH_WRITE }.
The default value is RAPI_RELOAD_WITH_WRITE.

(R8C)

• **When timer C is used (RAPI_TIMER_C specified)**

Specify a pointer to the array in which the comparison value is stored.

[0]: Specify comparison value 0 in 16 bits.

[1]: Specify comparison value 1 in 16 bits.

• **When timer RD is used (RAPI_TIMER_RD0 to RAPI_TIMER_RD4 specified)**

Specify a pointer to the array in which the comparison value is stored.

[0]: Specify the output operation of TRDIOAi (i=0,1).

[1]: Specify the output operation of TRDIOBi (i=0,1).

[2]: Specify the output operation of TRDIOCi (i=0,1).

[3]: Specify the output operation of TRDIODi (i=0,1).

[4]: Specify the comparison value of general register A in 16 bits.

[5]: Specify the comparison value of general register B in 16 bits.

[6]: Specify the comparison value of general register C in 16 bits.

[7]: Specify the comparison value of general register D in 16 bits.

To specify output operation for each output compare of array elements, the following definition values can be used. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. For elements of output operation corresponding to unused channels, set 0.

RAPI_OUT_0	Selects 0 output for the output waveform.
RAPI_OUT_1	Selects 1 output for the output waveform.
RAPI_OUT_TOGGLE	Selects toggle output for the output waveform.
RAPI_OUT_INIT_0	Selects 0 output for the initial output.
RAPI_OUT_INIT_1	Selects 1 output for the initial output.

• **When timer RE is used (RAPI_TIMER_RE specified)**

Specify a pointer to the array in which the comparison value is stored.

Specify comparison value of timer RE compare data register in 8 bits.

(H8/300H)

Specify a pointer to the array in which the comparison value is stored.

[0]: Specify the output operation of FTIOA pin.

[1]: Specify the output operation of FTIOB pin.

[2]: Specify the output operation of FTIOC pin.

[3]: Specify the output operation of FTIOD pin.

[4]: Specify the comparison value of general register A in 16 bits.

[5]: Specify the comparison value of general register B in 16 bits.

[6]: Specify the comparison value of general register C in 16 bits.

[7]: Specify the comparison value of general register D in 16 bits.

To specify output operation for each output compare of array elements, the following definition values can be used. To specify multiple definition values at the same time, use the symbol “|” to separate each specified value. For elements of output operation corresponding to unused channels, set 0.

RAPI_OUT_0	Selects 0 output for the output waveform.
RAPI_OUT_1	Selects 1 output for the output waveform.
RAPI_OUT_TOGGLE	Selects toggle output for the output waveform.
RAPI_OUT_INIT_0	Selects 0 output for the initial output.
RAPI_OUT_INIT_1	Selects 1 output for the initial output.

[data4]

(M16C)

Specify a pointer to the array in which the set value for each register of timer S is stored.

[0]: Specify the set value for the facility select and facility enable register.

Specify the channel for which the waveform generation facility is enabled.

[1]: Specify the set value for interrupt enable register 0.

Specify the channel for which IO/CO interrupt 0 request is enabled.

[2]: Specify the set value for interrupt enable register 1.

Specify the channel for which IO/CO interrupt 1 request is enabled.

[3]: Specify the set value for the count source divide-by-n register. Specify the value of 'n' in the formula “count source divided by (n + 1)” in 8 bits.

For the channels to be specified in each array element, use the following definition values. To specify multiple definition values at the same time, use the symbol “|” to separate each specified value. If 0 is specified, the value 0 is set in the corresponding register.

RAPI_CHANNEL0	Selects channel 0.
RAPI_CHANNEL1	Selects channel 1.
RAPI_CHANNEL2	Selects channel 2.
RAPI_CHANNEL3	Selects channel 3.
RAPI_CHANNEL4	Selects channel 4.
RAPI_CHANNEL5	Selects channel 5.
RAPI_CHANNEL6	Selects channel 6.
RAPI_CHANNEL7	Selects channel 7.

(R8C) (H8/300H)

Specify 0.

[data5]

(M16C)

Specify a pointer to the array in which the callback function is stored.

[0]: Specify a pointer to the callback function for IC/OC base timer interrupt.

If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for IC/OC interrupt 0.

If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for IC/OC interrupt 1.

If this pointer is not specified, 0 is set.

(R8C)

• **When timer C is used (RAPI_TIMER_C specified)**

Specify a pointer to the array in which the callback function is stored.

[0]: Specify a pointer to the callback function for timer C interrupt.

If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for compare match interrupt 0.

If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for compare match interrupt 1.

If this pointer is not specified, 0 is set.

• **When timer RD is used (RAPI_TIMER_RD0 to RAPI_TIMER_RD4 specified)**

Specify a pointer to the array in which the callback function is stored.

If this pointer is not specified, RAPI_NULL is set.

• **When timer RE is used (RAPI_TIMER_RE specified)**

Specify a pointer to the array in which the callback function is stored.

If this pointer is not specified, RAPI_NULL is set.

(H8/300H)

Specify a pointer to the variable in which the callback function is stored. If this pointer is not specified, RAPI_NULL is set.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (output compare mode)

Reference

[__EnableOutputCompare](#), [__DestroyOutputCompare](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- If when using the H8/300H an undefined value is specified in the second argument, program operation cannot be guaranteed.
- If when using the M16C an undefined value is specified for any waveform generation control register in the third argument, operation of the API cannot be guaranteed.
- If when using the H8/300H an undefined value is specified for the output operation of any channel in the third argument, operation of the API cannot be guaranteed.
- If when using the M16C or the H8/300H an undefined value is specified in the fourth argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```

#include "rapi_timer_r8c_13.h"

void TimerIntFunc0( void ){}
void TimerIntFunc1( void ){}
void TimerIntFunc2( void ){}

void func( void )
{
    unsigned int *p_func[] = {(void*) TimerIntFunc0, (void*) TimerIntFunc1,
                              (void*) TimerIntFunc2};
    unsigned char p_ic[] = {1,2,3};
    unsigned int p_cmp[] = {0x1234, 0x9876};

    /* Set up timer C as output compare mode */
    __CreateOutputCompare(
RAPI_TIMER_C|RAPI_TIMER_ON|RAPI_RELOAD|RAPI_L_1|RAPI_L_0|RAPI_F32|
    RAPI_CMP02_ENABLE|RAPI_CMP12_ENABLE, p_cmp, p_ic, p_func );
}

```

__EnableOutputCompare

Synopsis

<Control operation of output compare mode>

Boolean __EnableOutputCompare(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of the timer that is set to specified output compare mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_S	Selects timer S.
RAPI_TIMER_ON	Sets the timer that is set to output compare mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to output compare mode to stop operating.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_ON	Sets the timer that is set to output compare mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to output compare mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.
RAPI_TIMER_ON	Sets the timer that is set to output compare mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to output compare mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (output compare mode)

Reference

[__CreateOutputCompare](#), [__DestroyOutputCompare](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be

guaranteed.

- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer C as output compare mode */
    __EnableOutputCompare( RAPI_TIMER_C|RAPI_TIMER_ON );
}
```

__DestroyOutputCompare

Synopsis

<Discard settings of output compare mode>

Boolean **__DestroyOutputCompare(unsigned long data)**

data	Setup data (content differs with MCU type)
------	--

Description

Discards settings of the timer that is set to specified output compare mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_S	Selects timer S.
--------------	------------------

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (output compare mode)

Reference

[__CreateOutputCompare](#), [__EnableOutputCompare](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"
```

```
void func( void )
{
    /* Destroy the setting of timer C as output compare mode */
    __DestroyOutputCompare( RAPI_TIMER_C );
}
```


__SetTimerRegister

Synopsis

<Set timer register>

Boolean __SetTimerRegister(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which register value is stored

Description

Sets the registers of a specified timer.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.

[data2]

The content of a pointer to the buffer in which the register value is stored must be specified as described below. The value is set in each register in order of buffer pointer elements.

(M16C)

• When using timer A (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)

- [0]: Specify the set value for the timer Ai mode register (i = 0–4).
- [1]: Specify the set value for the timer Ai register (i = 0–4).
- [2]: Specify the set value for the up/down flag register.
- [3]: Specify the set value for the one-shot start flag register.
- [4]: Specify the set value for the trigger select register.
- [5]: Specify the set value for the time-clock prescaler reset register.
- [6]: Specify the set value for the count start flag register.

• When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2)

- [0]: Specify the set value for the timer Bi mode register (i = 0–2).
- [1]: Specify the set value for the timer Bi register (i = 0–2).
- [3]: Specify the set value for the time-clock prescaler reset register.
- [4]: Specify the set value for the count start flag register.

• When using timer S (RAPI_TIMER_S specified)

- [0]: Specify the set value for the base timer register.
- [1]: Specify the set value for the base timer reset register.
- [2]: Specify the set value for base timer control register 0.
- [3]: Specify the set value for base timer control register 1.
- [4]: Specify the set value for the count source divide-by-n register.
- [5]: Specify the set value for time measurement control register 0.
- [6]: Specify the set value for time measurement control register 1.
- [7]: Specify the set value for time measurement control register 2.
- [8]: Specify the set value for time measurement control register 3.
- [9]: Specify the set value for time measurement control register 4.
- [10]: Specify the set value for time measurement control register 5.
- [11]: Specify the set value for time measurement control register 6.
- [12]: Specify the set value for time measurement control register 7.
- [13]: Specify the set value for time measurement prescaler register 6.
- [14]: Specify the set value for time measurement prescaler register 7.
- [15]: Specify the set value for waveform generation control register 0.
- [16]: Specify the set value for waveform generation control register 1.
- [17]: Specify the set value for waveform generation control register 2.
- [18]: Specify the set value for waveform generation control register 3.
- [19]: Specify the set value for waveform generation control register 4.
- [20]: Specify the set value for waveform generation control register 5.
- [21]: Specify the set value for waveform generation control register 6.
- [22]: Specify the set value for waveform generation control register 7.
- [23]: Specify the set value for waveform generation register 0.
- [24]: Specify the set value for waveform generation register 1.
- [25]: Specify the set value for waveform generation register 2.
- [26]: Specify the set value for waveform generation register 3.

- [27]: Specify the set value for waveform generation register 4.
- [28]: Specify the set value for waveform generation register 5.
- [29]: Specify the set value for waveform generation register 6.
- [30]: Specify the set value for waveform generation register 7.
- [31]: Specify the set value for the facility select register.
- [32]: Specify the set value for the facility enable register.
- [33]: Specify the set value for the interrupt request register.
- [34]: Specify the set value for interrupt enable register 0.
- [35]: Specify the set value for interrupt enable register 1.

(R8C)

- **When using timer C (RAPI_TIMER_C specified)**

- [0]: Specify the set value for the timer C output control register.
- [1]: Specify the set value for timer C control register 0.
- [2]: Specify the set value for timer C control register 1.
- [3]: Specify the set value for the capture & compare 0 register.
- [4]: Specify the set value for the compare 1 register.

- **When using timer X (RAPI_TIMER_X specified)**

- [0]: Specify the set value for the timer count source setup register.
- [1]: Specify the set value for the prescaler X register.
- [2]: Specify the set value for the timer X register.
- [3]: Specify the set value for the timer X mode register.

- **When using timer Y (RAPI_TIMER_Y specified)**

- [0]: Specify the set value for the timer count source setup register.
- [1]: Specify the set value for the prescaler Y register.
- [2]: Specify the set value for the timer Y primary register.
- [3]: Specify the set value for the timer Y secondary register.
- [4]: Specify the set value for the timer Y & Z waveform output control register.
- [5]: Specify the set value for the timer Y & Z output control register.
- [6]: Specify the set value for the timer Y & Z mode register.

- **When using timer Z (RAPI_TIMER_Z specified)**

- [0]: Specify the set value for the timer count source setup register.
- [1]: Specify the set value for the prescaler Z register.
- [2]: Specify the set value for the timer Z primary register.
- [3]: Specify the set value for the timer Z secondary register.
- [4]: Specify the set value for the timer Y & Z waveform output control register.
- [5]: Specify the set value for the timer Y & Z output control register.
- [6]: Specify the set value for the timer Y & Z mode register.

- **When using timer RA (RAPI_TIMER_RA specified)**

- [0]: Specify the set value for the timer RA I/O control register.
- [1]: Specify the set value for the timer RA prescaler register.
- [2]: Specify the set value for the timer RA register.
- [3]: Specify the set value for the timer RA mode register.
- [4]: Specify the set value for the timer RA control register.

- **When using timer RB (RAPI_TIMER_RB specified)**

- [0]: Specify the set value for the timer RB one shot control register.

[1]: Specify the set value for the timer RB I/O control register.

[2]: Specify the set value for the timer RB prescaler register.

[3]: Specify the set value for the timer RB primary register.

[4]: Specify the set value for the timer RB secondary register.

[5]: Specify the set value for the timer RB mode register.

[6]: Specify the set value for the timer RB control register.

• **When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD1)**

[0]: Specify the set value for the timer RD mode register.

[1]: Specify the set value for the timer RD PWM mode register.

[2]: Specify the set value for the timer RD function control register.

[3]: Specify the set value for the timer RD general register Ai (i=0,1).

[4]: Specify the set value for the timer RD general register Bi (i=0,1).

[5]: Specify the set value for the timer RD general register Ci (i=0,1).

[6]: Specify the set value for the timer RD general register Di (i=0,1).

[7]: Specify the set value for the timer RD digital filter function select register i(i=0,1).

[8]: Specify the set value for the timer RD control register i(i=0,1).

[9]: Specify the set value for the timer RD I/O control register Ai(i=0,1).

[10]: Specify the set value for the timer RD I/O control register Ci(i=0,1).

[11]: Specify the set value for the timer RD status register i(i=0,1).

[12]: Specify the set value for the timer RD interrupt enable register i(i=0,1).

[13]: Specify the set value for the timer RD counter i(i=0,1).

[14]: Specify the set value for the timer RD start register.

[15]: Specify the set value for the timer RD output master enable register 1.

[16]: Specify the set value for the timer RD output master enable register 2.

[17]: Specify the set value for the timer RD output control register.

• **When using timer RE (RAPI_TIMER_RE)**

[0]: Specify the set value for the timer RE second data register.

[1]: Specify the set value for the timer RE minute data register.

[2]: Specify the set value for the timer RE hour data register.

[3]: Specify the set value for the timer RE day of week data register.

[4]: Specify the set value for the timer RE control register 2.

[5]: Specify the set value for the timer RE count source select register.

[6]: Specify the set value for the timer RE control register 1.

(H8/300H)

• **When using timer A (RAPI_TIMER_A specified)**

[0]: Specify the set value for timer mode register A.

[1]: Specify the set value for timer counter A.

• **When using timer B1 (RAPI_TIMER_B1 specified)**

[0]: Specify the set value for timer mode register B1.

[1]: Specify the set value for timer load register B1.

• **When using timer V (RAPI_TIMER_V specified)**

[0]: Specify the set value for timer counter V.

[1]: Specify the set value for time constant register A.

[2]: Specify the set value for time constant register B.

[3]: Specify the set value for timer control register V0.

[4]: Specify the set value for timer control register V1.

[5]: Specify the set value for timer control/status register V.

• **When using timer W (RAPI_TIMER_W specified)**

[0]: Specify the set value for timer control register W.

[1]: Specify the set value for timer interrupt enable register W.

[2]: Specify the set value for timer status register W.

[3]: Specify the set value for timer I/O control register 0.

[4]: Specify the set value for timer I/O control register 1.

[5]: Specify the set value for the timer counter.

[6]: Specify the set value for general register A.

[7]: Specify the set value for general register B.

[8]: Specify the set value for general register C.

[9]: Specify the set value for general register D.

[10]: Specify the set value for the timer mode register W.

• **When using timer Z (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)**

[0]: Specify the set value for the timer mode register.

[1]: Specify the set value for the timer PWM mode register.

[2]: Specify the set value for the timer function control register.

[3]: Specify the set value for the timer output master enable register.

[4]: Specify the set value for the timer output control register.

[5]: Specify the set value for the timer counter.

[6]: Specify the set value for general register A $i(i=0,1)$.

[7]: Specify the set value for general register B $i(i=0,1)$.

[8]: Specify the set value for general register C $i(i=0,1)$.

[9]: Specify the set value for general register D $i(i=0,1)$.

[10]: Specify the set value for timer control register $_i$ ($i = 0, 1$).

[11]: Specify the set value for timer I/O control register A_i ($i = 0, 1$).

[12]: Specify the set value for timer I/O control register C_i ($i = 0, 1$).

[13]: Specify the set value for timer status register $_i$ ($i = 0, 1$).

[14]: Specify the set value for timer interrupt enable register $_i$ ($i = 0, 1$).

[15]: Specify the set value for PWM mode output level control register $_i$ ($i = 0, 1$).

[16]: Specify the set value for the timer start register.

• **When using timer RC (RAPI_TIMER_RC)**

[0]: Specify the set value for timer RC control register 1.

[1]: Specify the set value for timer RC control register 2.

[2]: Specify the set value for timer RC interrupt enable register.

[3]: Specify the set value for timer RC status register.

[4]: Specify the set value for timer RC I/O control register 0.

[5]: Specify the set value for timer RC I/O control register 1.

[6]: Specify the set value for timer RC output enable register.

[7]: Specify the set value for timer RC digital filter function select register.

[8]: Specify the set value for timer RC counter.

[9]: Specify the set value for general register A.

[10]: Specify the set value for general register B.

- [11]: Specify the set value for general register C.
- [12]: Specify the set value for general register D.
- [13]: Specify the set value for timer RC mode register.
- **When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD3 specified)**
- [0]: Specify the set value for the timer RD mode register.
- [1]: Specify the set value for the timer RD PWM mode register.
- [2]: Specify the set value for the timer RD function control register.
- [3]: Specify the set value for the timer RD output master enable register 1.
- [4]: Specify the set value for the timer RD output master enable register 2.
- [5]: Specify the set value for the timer RD output control register.
- [6]: Specify the set value for the timer RD counter $i(i=0,1)$.
- [7]: Specify the set value for general register $A_i(i=0,1)$.
- [8]: Specify the set value for general register $B_i(i=0,1)$.
- [9]: Specify the set value for general register $C_i(i=0,1)$.
- [10]: Specify the set value for general register $D_i(i=0,1)$.
- [11]: Specify the set value for timer RD control register $_i$ ($i = 0, 1$).
- [12]: Specify the set value for timer RD I/O control register A_i ($i = 0, 1$).
- [13]: Specify the set value for timer RD I/O control register C_i ($i = 0, 1$).
- [14]: Specify the set value for timer RD status register $_i$ ($i = 0, 1$).
- [15]: Specify the set value for timer RD interrupt enable register $_i$ ($i = 0, 1$).
- [16]: Specify the set value for PWM mode output level control register $_i$ ($i = 0, 1$).
- [17]: Specify the set value for the timer RD digital filter function select register i ($i = 0, 1$).
- [18]: Specify the set value for the timer RD start register.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (register manipulation)
Reference	__EnableTimerRegister , __ClearTimerRegister , __GetTimerRegister
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.

Program example

```
#include " rapi_timer_r8c_13.h"

void func( void )
{
    unsigned char data[] = {0,0,0,0,0,0,0};

    /* Set up timer Z register */
    __SetTimerRegister( RAPI_TIMER_Z, data );
}
```

__EnableTimerRegister

Synopsis

<Control operation of timer register>

Boolean __EnableTimerRegister(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Controls operation of a specified timer by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.
RAPI_TIMER_ON	Sets the selected timer to start operating.
RAPI_TIMER_OFF	Sets the selected timer to stop operating.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_ON	Sets the selected timer to start operating.
RAPI_TIMER_OFF	Sets the selected timer to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.

RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.
RAPI_TIMER_ON	Sets the selected timer to start operating.
RAPI_TIMER_OFF	Sets the selected timer to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (register manipulation)

Reference

[__SetTimerRegister](#), [__ClearTimerRegister](#), [__GetTimerRegister](#)

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- To specify commencement of timer operation when using timer V of the H8/300H, set the clock and count condition to be supplied to TCNTV that are specified in __SetTimerRegister immediately preceding this API.

Program example

```
#include " rapi_timer_r8c_13.h"

void func( void )
{
    /* Activate timer C */
    __EnableTimerRegister( RAPI_TIMER_C|RAPI_TIMER_ON );
}
```


__ClearTimerRegister

Synopsis

<Clear timer register>

Boolean __ClearTimerRegister(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Sets the timer register of a specified timer to its initial value after reset.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (register manipulation)
Reference	__SetTimerRegister , __EnableTimerRegister , __GetTimerRegister
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.
Program example	

```
#include " rapi_timer_r8c_13.h"

void func( void )
{
    /* Clear the setting of timer C */
    __ClearTimerRegister( RAPI_TIMER_C );
}
```

__GetTimerRegister

Synopsis

<Get timer register value>

Boolean __GetTimerRegister(unsigned long data1, unsigned int *data2)

data1	Setup data (content differs with MCU type)
data2	Pointer to the buffer in which register value is stored

Description

Gets the counter value of a specified timer.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.

[data2]

Specify a pointer to the array in which the acquired register value is stored.
The content of the array is described below.

(M16C)

- **When using timer A (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)**

- [0]: Store the value of timer Ai mode register (i = 0–4).
- [1]: Store the value of timer Ai register (i = 0–4).
- [2]: Store the value of the up/down flag register.
- [3]: Store the value of the one-shot start flag register.
- [4]: Store the value of the trigger select register.
- [5]: Store the value of the time-clock prescaler reset flag register.
- [6]: Store the value of the count start flag register.

- **When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)**

- [0]: Store the value of timer Bi mode register (i = 0–2).
- [1]: Store the value of timer Bi register (i = 0–2).
- [2]: Store the value of the time-clock prescaler reset flag register.
- [3]: Store the value of the count start flag register.

- **When using timer S (RAPI_TIMER_S specified)**

- [0]: Store the value of the base timer register.
- [1]: Store the value of the base timer reset register.
- [2]: Store the value of base timer control register 0.
- [3]: Store the value of base timer control register 1.
- [4]: Store the value of the count source divide-by-n register.
- [5]: Store the value of time measurement control register 0.
- [6]: Store the value of time measurement control register 1.
- [7]: Store the value of time measurement control register 2.
- [8]: Store the value of time measurement control register 3.
- [9]: Store the value of time measurement control register 4.
- [10]: Store the value of time measurement control register 5.
- [11]: Store the value of time measurement control register 6.
- [12]: Store the value of time measurement control register 7.
- [13]: Store the value of time measurement prescaler register 6.
- [14]: Store the value of time measurement prescaler register 7.
- [15]: Store the value of waveform generation control register 0.
- [16]: Store the value of waveform generation control register 1.
- [17]: Store the value of waveform generation control register 2.
- [18]: Store the value of waveform generation control register 3.
- [19]: Store the value of waveform generation control register 4.
- [20]: Store the value of waveform generation control register 5.
- [21]: Store the value of waveform generation control register 6.
- [22]: Store the value of waveform generation control register 7.
- [23]: Store the value of time measurement register 0.
- [24]: Store the value of time measurement register 1/waveform generation register 1.
- [25]: Store the value of time measurement register 2/waveform generation register 2.

- [26]: Store the value of time measurement register 3/waveform generation register 3.
- [27]: Store the value of time measurement register 4/waveform generation register 4.
- [28]: Store the value of time measurement register 5/waveform generation register 5.
- [29]: Store the value of time measurement register 6/waveform generation register 6.
- [30]: Store the value of time measurement register 7/waveform generation register 7.
- [31]: Store the value of the facility select register.
- [32]: Store the value of the facility enable register.
- [33]: Store the value of the interrupt request register.
- [34]: Store the value of interrupt enable register 0.
- [35]: Store the value of interrupt enable register 1.

(R8C)

- **When using timer C (RAPI_TIMER_C specified)**

- [0]: Store the value of the timer C register.
- [1]: Store the value of the capture & compare 0 register.
- [2]: Store the value of the compare 1 register.
- [3]: Store the value of the timer C output control register.
- [4]: Store the value of timer C control register 1.
- [5]: Store the value of timer C control register 0.

- **When using timer X (RAPI_TIMER_X specified)**

- [0]: Store the value of the timer count source setup register.
- [1]: Store the value of the prescaler X register.
- [2]: Store the value of the timer X register.
- [3]: Store the value of the timer X mode register.

- **When using timer Y (RAPI_TIMER_Y specified)**

- [0]: Store the value of the timer count source setup register.
- [1]: Store the value of the prescaler Y register.
- [2]: Store the value of the timer Y primary register.
- [3]: Store the value of the timer Y secondary register.
- [4]: Store the value of the timer Y & Z waveform output control register.
- [5]: Store the value of the timer Y & Z output control register.
- [6]: Store the value of the timer Y & Z mode register.

- **When using timer Z (RAPI_TIMER_Z specified)**

- [0]: Store the value of the timer count source setup register.
- [1]: Store the value of the prescaler Z register.
- [2]: Store the value of the timer Z primary register.
- [3]: Store the value of the timer Z secondary register.
- [4]: Store the value of the timer Y & Z waveform output control register.
- [5]: Store the value of the timer Y & Z output control register.
- [6]: Store the value of the timer Y & Z mode register.

- **When using timer RA (RAPI_TIMER_RA specified)**

- [0]: Specify the set value for the timer RA I/O control register.
- [1]: Specify the set value for the timer RA prescaler register.
- [2]: Specify the set value for the timer RA register.
- [3]: Specify the set value for the timer RA mode register.
- [4]: Specify the set value for the timer RA control register.

• **When using timer RB (RAPI_TIMER_RB specified)**

- [0]: Specify the set value for the timer RB one shot control register.
- [1]: Specify the set value for the timer RB I/O control register.
- [2]: Specify the set value for the timer RB prescaler register.
- [3]: Specify the set value for the timer RB primary register.
- [4]: Specify the set value for the timer RB secondary register.
- [5]: Specify the set value for the timer RB mode register.
- [6]: Specify the set value for the timer RB control register.

• **When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD1)**

- [0]: Specify the set value for the timer RD mode register.
- [1]: Specify the set value for the timer RD PWM mode register.
- [2]: Specify the set value for the timer RD function control register.
- [3]: Specify the set value for the timer RD general register Ai (i=0,1).
- [4]: Specify the set value for the timer RD general register Bi (i=0,1).
- [5]: Specify the set value for the timer RD general register Ci (i=0,1).
- [6]: Specify the set value for the timer RD general register Di (i=0,1).
- [7]: Specify the set value for the timer RD digital filter function select register i(i=0,1).
- [8]: Specify the set value for the timer RD control register i(i=0,1).
- [9]: Specify the set value for the timer RD I/O control register Ai(i=0,1).
- [10]: Specify the set value for the timer RD I/O control register Ci(i=0,1).
- [11]: Specify the set value for the timer RD status register i(i=0,1).
- [12]: Specify the set value for the timer RD interrupt enable register i(i=0,1).
- [13]: Specify the set value for the timer RD counter i(i=0,1).
- [14]: Specify the set value for the timer RD start register.
- [15]: Specify the set value for the timer RD output master enable register 1.
- [16]: Specify the set value for the timer RD output master enable register 2.
- [17]: Specify the set value for the timer RD output control register.

• **When using timer RE (RAPI_TIMER_RE)**

- [0]: Specify the set value for the timer RE second data register.
- [1]: Specify the set value for the timer RE minute data register.
- [2]: Specify the set value for the timer RE hour data register.
- [3]: Specify the set value for the timer RE day of week data register.
- [4]: Specify the set value for the timer RE control register 2.
- [5]: Specify the set value for the timer RE count source select register.
- [6]: Specify the set value for the timer RE control register 1.

(H8/300H)

• **When using timer A (RAPI_TIMER_A specified)**

- [0]: Store the value of timer mode register A.
- [1]: Store the value of timer counter A.

• **When using timer B1 (RAPI_TIMER_B1 specified)**

- [0]: Store the value of timer mode register B1.
- [1]: Store the value of timer counter B1.

• **When using timer V (RAPI_TIMER_V specified)**

- [0]: Store the value of timer counter V.

- [1]: Store the value of time constant register A.
- [2]: Store the value of time constant register B.
- [3]: Store the value of timer control register V0.
- [4]: Store the value of timer control register V1.
- [5]: Store the value of timer control/status register V.

• **When using timer W (RAPI_TIMER_W specified)**

- [0]: Store the value of timer mode register W.
- [1]: Store the value of timer control register W.
- [2]: Store the value of timer interrupt master enable register W.
- [3]: Store the value of timer status register W.
- [4]: Store the value of timer I/O control register 0.
- [5]: Store the value of timer I/O control register 1.
- [6]: Store the value of the timer counter.
- [7]: Store the value of general register A.
- [8]: Store the value of general register B.
- [9]: Store the value of general register C.
- [10]: Store the value of general register D.

• **When using timer Z (RAPI_TIMER_Z specified)**

- [0]: Store the value of the timer start register.
- [1]: Store the value of the timer mode register.
- [2]: Store the value of the timer PWM mode register.
- [3]: Store the value of the timer function control register.
- [4]: Store the value of the timer output master enable register.
- [5]: Store the value of the timer output control register.
- [6]: Store the value of timer counter_i (i = 0, 1).
- [7]: Store the value of general register A_i (i = 0, 1).
- [8]: Store the value of general register B_i (i=0, 1).
- [9]: Store the value of general register C_i (i=0, 1).
- [10]: Store the value of general register D_i (i=0, 1).
- [11]: Store the value of timer control register_i (i = 0, 1).
- [12]: Store the value of timer I/O control register A_i (i = 0, 1).
- [13]: Store the value of timer I/O control register B_i (i = 0, 1).
- [14]: Store the value of timer status register_i (i = 0, 1).
- [15]: Store the value of timer interrupt enable register_i (i = 0, 1).
- [16]: Store the value of PWM mode output level control register_i (i = 0, 1).

• **When using timer RC (RAPI_TIMER_RC)**

- [0]: Specify the set value for timer RC control register 1.
- [1]: Specify the set value for timer RC control register 2.
- [2]: Specify the set value for timer RC interrupt enable register.
- [3]: Specify the set value for timer RC status register.
- [4]: Specify the set value for timer RC I/O control register 0.
- [5]: Specify the set value for timer RC I/O control register 1.
- [6]: Specify the set value for timer RC output enable register.
- [7]: Specify the set value for timer RC digital filter function select register.
- [8]: Specify the set value for timer RC counter.

- [9]: Specify the set value for general register A.
- [10]: Specify the set value for general register B.
- [11]: Specify the set value for general register C.
- [12]: Specify the set value for general register D.
- [13]: Specify the set value for timer RC mode register.
- **When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD3 specified)**
- [0]: Specify the set value for the timer RD mode register.
- [1]: Specify the set value for the timer RD PWM mode register.
- [2]: Specify the set value for the timer RD function control register.
- [3]: Specify the set value for the timer RD output master enable register 1.
- [4]: Specify the set value for the timer RD output master enable register 2.
- [5]: Specify the set value for the timer RD output control register.
- [6]: Specify the set value for the timer RD counter i(i=0,1).
- [7]: Specify the set value for general register A_i(i=0,1).
- [8]: Specify the set value for general register B_i(i=0,1).
- [9]: Specify the set value for general register C_i(i=0,1).
- [10]: Specify the set value for general register D_i(i=0,1).
- [11]: Specify the set value for timer RD control register_i (i = 0, 1).
- [12]: Specify the set value for timer RD I/O control register A_i (i = 0, 1).
- [13]: Specify the set value for timer RD I/O control register C_i (i = 0, 1).
- [14]: Specify the set value for timer RD status register_i (i = 0, 1).
- [15]: Specify the set value for timer RD interrupt enable register_i (i = 0, 1).
- [16]: Specify the set value for PWM mode output level control register_i (i = 0, 1).
- [17]: Specify the set value for the timer RD digital filter function select register i(i = 0, 1).
- [18]: Specify the set value for the timer RD start register.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (register manipulation)
Reference	__SetTimerRegister , __EnableTimerRegister , __ClearTimerRegister
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.

Program example	<pre>#include " rapi_timer_r8c_13.h" void func(void) { unsigned int data[7]; /* Get the value of timer Z registers */ __GetTimerRegister(RAPI_TIMER_Z, data); }</pre>
------------------------	---

4.2.3 I/O Port __SetIOPort

Synopsis

<Set I/O port>

Boolean __SetIOPort(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)

Description

Sets the operating conditions of a specified I/O port.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value. Note, however, that multiple ports cannot be specified at the same time.

(M16C)

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_0_0	Port P0 ₀	RAPI_PORT_0_1	Port P0 ₁
RAPI_PORT_0_2	Port P0 ₂	RAPI_PORT_0_3	Port P0 ₃
RAPI_PORT_0_4	Port P0 ₄	RAPI_PORT_0_5	Port P0 ₅
RAPI_PORT_0_6	Port P0 ₆	RAPI_PORT_0_7	Port P0 ₇
RAPI_PORT_1_0	Port P1 ₀	RAPI_PORT_1_1	Port P1 ₁
RAPI_PORT_1_2	Port P1 ₂	RAPI_PORT_1_3	Port P1 ₃
RAPI_PORT_1_4	Port P1 ₄	RAPI_PORT_1_5	Port P1 ₅
RAPI_PORT_1_6	Port P1 ₆	RAPI_PORT_1_7	Port P1 ₇
RAPI_PORT_2_0	Port P2 ₀	RAPI_PORT_2_1	Port P2 ₁
RAPI_PORT_2_2	Port P2 ₂	RAPI_PORT_2_3	Port P2 ₃
RAPI_PORT_2_4	Port P2 ₄	RAPI_PORT_2_5	Port P2 ₅
RAPI_PORT_2_6	Port P2 ₆	RAPI_PORT_2_7	Port P2 ₇
RAPI_PORT_3_0	Port P3 ₀	RAPI_PORT_3_1	Port P3 ₁
RAPI_PORT_3_2	Port P3 ₂	RAPI_PORT_3_3	Port P3 ₃
RAPI_PORT_3_4	Port P3 ₄	RAPI_PORT_3_5	Port P3 ₅
RAPI_PORT_3_6	Port P3 ₆	RAPI_PORT_3_7	Port P3 ₇
RAPI_PORT_6_0	Port P6 ₀	RAPI_PORT_6_1	Port P6 ₁
RAPI_PORT_6_2	Port P6 ₂	RAPI_PORT_6_3	Port P6 ₃
RAPI_PORT_6_4	Port P6 ₄	RAPI_PORT_6_5	Port P6 ₅
RAPI_PORT_6_6	Port P6 ₆	RAPI_PORT_6_7	Port P6 ₇
RAPI_PORT_7_0	Port P7 ₀	RAPI_PORT_7_1	Port P7 ₁
RAPI_PORT_7_2	Port P7 ₂	RAPI_PORT_7_3	Port P7 ₃
RAPI_PORT_7_4	Port P7 ₄	RAPI_PORT_7_5	Port P7 ₅
RAPI_PORT_7_6	Port P7 ₆	RAPI_PORT_7_7	Port P7 ₇
RAPI_PORT_8_0	Port P8 ₀	RAPI_PORT_8_1	Port P8 ₁
RAPI_PORT_8_2	Port P8 ₂	RAPI_PORT_8_3	Port P8 ₃
RAPI_PORT_8_4	Port P8 ₄	RAPI_PORT_8_5	Port P8 ₅
RAPI_PORT_8_6	Port P8 ₆	RAPI_PORT_8_7	Port P8 ₇

RAPI_PORT_9_0	Port P9 ₀	RAPI_PORT_9_1	Port P9 ₁
RAPI_PORT_9_2	Port P9 ₂	RAPI_PORT_9_3	Port P9 ₃
RAPI_PORT_9_5	Port P9 ₅	RAPI_PORT_9_6	Port P9 ₆
RAPI_PORT_9_7	Port P9 ₇	RAPI_PORT_10_0	Port P10 ₀
RAPI_PORT_10_1	Port P10 ₁	RAPI_PORT_10_2	Port P10 ₂
RAPI_PORT_10_3	Port P10 ₃	RAPI_PORT_10_4	Port P10 ₄
RAPI_PORT_10_5	Port P10 ₅	RAPI_PORT_10_6	Port P10 ₆
RAPI_PORT_10_7	Port P10 ₇		

The definition values related to port settings are described below.

RAPI_PORT_INPUT	Sets a selected port for input.
RAPI_PORT_OUTPUT	Sets a selected port for output.
RAPI_PULLED_HIGH	Sets a selected port to be pulled high.
RAPI_NOT_PULLED_HIGH	Sets a selected port not to be pulled high.
RAPI_LATCH	Sets a selected port to read the port latch regardless of whether it is set for input or output. Specifiable only when port P1 is used.

(R8C)

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_0_0	Port P0 ₀	RAPI_PORT_0_1	Port P0 ₁
RAPI_PORT_0_2	Port P0 ₂	RAPI_PORT_0_3	Port P0 ₃
RAPI_PORT_0_4	Port P0 ₄	RAPI_PORT_0_5	Port P0 ₅
RAPI_PORT_0_6	Port P0 ₆	RAPI_PORT_0_7	Port P0 ₇
RAPI_PORT_1_0	Port P1 ₀	RAPI_PORT_1_1	Port P1 ₁
RAPI_PORT_1_2	Port P1 ₂	RAPI_PORT_1_3	Port P1 ₃
RAPI_PORT_1_4	Port P1 ₄	RAPI_PORT_1_5	Port P1 ₅
RAPI_PORT_1_6	Port P1 ₆	RAPI_PORT_1_7	Port P1 ₇
RAPI_PORT_2_0	Port P2 ₀	RAPI_PORT_2_1	Port P2 ₁
RAPI_PORT_2_2	Port P2 ₂	RAPI_PORT_2_3	Port P2 ₃
RAPI_PORT_2_4	Port P2 ₄	RAPI_PORT_2_5	Port P2 ₅
RAPI_PORT_2_6	Port P2 ₆	RAPI_PORT_2_7	Port P2 ₇
RAPI_PORT_3_0	Port P3 ₀	RAPI_PORT_3_1	Port P3 ₁
RAPI_PORT_3_2	Port P3 ₂	RAPI_PORT_3_3	Port P3 ₃
RAPI_PORT_3_4	Port P3 ₄	RAPI_PORT_3_5	Port P3 ₅
RAPI_PORT_3_7	Port P3 ₇	RAPI_PORT_4_3	Port P4 ₃
RAPI_PORT_4_4	Port P4 ₄	RAPI_PORT_4_5	Port P4 ₅
RAPI_PORT_6_0	Port P6 ₀	RAPI_PORT_6_1	Port P6 ₁
RAPI_PORT_6_2	Port P6 ₂	RAPI_PORT_6_3	Port P6 ₃
RAPI_PORT_6_4	Port P6 ₄	RAPI_PORT_6_5	Port P6 ₅
RAPI_PORT_6_6	Port P6 ₆	RAPI_PORT_6_7	Port P6 ₇

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_INPUT	Sets a selected port for input.
RAPI_PORT_OUTPUT	Sets a selected port for output.
RAPI_PULLED_HIGH	Sets a selected port to be pulled high.

RAPI_NOT_PULLED_HIGH	Sets a selected port not to be pulled high.
RAPI_DRIVE_CAPACITY_H	Sets the N-channel output transistor drive capacity of a selected port to High. Specifiable only when port P1 is used.
RAPI_DRIVE_CAPACITY_L	Sets the N-channel output transistor drive capacity of a selected port to Low. Specifiable only when port P1 is used.

(H8/300H)

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_4	Port P14
RAPI_PORT_1_5	Port P15	RAPI_PORT_1_6	Port P16
RAPI_PORT_1_7	Port P17	RAPI_PORT_2_0	Port P20
RAPI_PORT_2_1	Port P21	RAPI_PORT_2_2	Port P22
RAPI_PORT_2_3	Port P23	RAPI_PORT_2_4	Port P24
RAPI_PORT_2_5	Port P25	RAPI_PORT_2_6	Port P26
RAPI_PORT_2_7	Port P27	RAPI_PORT_3_0	Port P30
RAPI_PORT_3_1	Port P31	RAPI_PORT_3_2	Port P32
RAPI_PORT_3_3	Port P33	RAPI_PORT_3_4	Port P34
RAPI_PORT_3_5	Port P35	RAPI_PORT_3_6	Port P36
RAPI_PORT_3_7	Port P37	RAPI_PORT_5_0	Port P50
RAPI_PORT_5_1	Port P51	RAPI_PORT_5_2	Port P52
RAPI_PORT_5_3	Port P53	RAPI_PORT_5_4	Port P54
RAPI_PORT_5_5	Port P55	RAPI_PORT_5_6	Port P56
RAPI_PORT_5_7	Port P57	RAPI_PORT_6_0	Port P60
RAPI_PORT_6_1	Port P61	RAPI_PORT_6_2	Port P62
RAPI_PORT_6_3	Port P63	RAPI_PORT_6_4	Port P64
RAPI_PORT_6_5	Port P65	RAPI_PORT_6_6	Port P66
RAPI_PORT_6_7	Port P67	RAPI_PORT_7_0	Port P70
RAPI_PORT_7_1	Port P71	RAPI_PORT_7_2	Port P72
RAPI_PORT_7_4	Port P74	RAPI_PORT_7_5	Port P75
RAPI_PORT_7_6	Port P76	RAPI_PORT_7_7	Port P77
RAPI_PORT_8_0	Port P80	RAPI_PORT_8_1	Port P81
RAPI_PORT_8_2	Port P82	RAPI_PORT_8_3	Port P83
RAPI_PORT_8_4	Port P84	RAPI_PORT_8_5	Port P85
RAPI_PORT_8_6	Port P86	RAPI_PORT_8_7	Port P87
RAPI_PORT_9_0	Port P90	RAPI_PORT_9_1	Port P91
RAPI_PORT_9_2	Port P92	RAPI_PORT_9_3	Port P93
RAPI_PORT_9_4	Port P94	RAPI_PORT_9_5	Port P95
RAPI_PORT_9_6	Port P96	RAPI_PORT_9_7	Port P97
RAPI_PORT_C_0	Port PC0	RAPI_PORT_C_1	Port PC1
RAPI_PORT_C_2	Port PC2	RAPI_PORT_C_3	Port PC3
RAPI_PORT_D_0	Port PD0	RAPI_PORT_D_1	Port PD1
RAPI_PORT_D_2	Port PD2	RAPI_PORT_D_3	Port PD3
RAPI_PORT_D_4	Port PD4	RAPI_PORT_D_5	Port PD5

RAPI_PORT_D_6	PortPD6	RAPI_PORT_D_7	PortPD7
RAPI_PORT_E_0	PortPE0	RAPI_PORT_E_1	PortPE1
RAPI_PORT_E_2	PortPE2	RAPI_PORT_E_3	PortPE3
RAPI_PORT_E_4	PortPE4	RAPI_PORT_E_5	PortPE5
RAPI_PORT_E_6	PortPE6	RAPI_PORT_E_7	PortPE7
RAPI_PORT_F_0	PortPF0	RAPI_PORT_G_0	PortPG0
RAPI_PORT_G_1	PortPG1	RAPI_PORT_G_2	PortPG2
RAPI_PORT_G_3	PortPG3	RAPI_PORT_G_4	PortPG4
RAPI_PORT_G_5	PortPG5	RAPI_PORT_G_6	PortPG6
RAPI_PORT_G_7	PortPG7	RAPI_PORT_H_0	PortPH0
RAPI_PORT_H_1	PortPH1	RAPI_PORT_H_2	PortPH2
RAPI_PORT_H_3	PortPH3	RAPI_PORT_H_4	PortPH4
RAPI_PORT_H_5	PortPH5	RAPI_PORT_H_6	PortPH6
RAPI_PORT_H_7	PortPH7	RAPI_PORT_J_0	PortPJ0
RAPI_PORT_J_1	PortPJ1		

The definition values related to port settings are described below.

RAPI_PORT_INPUT	Sets a selected port for input.
RAPI_PORT_OUTPUT	Sets a selected port for output.
RAPI_PULLED_HIGH	Sets a selected port to be pulled high.
RAPI_NOT_PULLED_HIGH	Sets a selected port not to be pulled high.

[data2]

(M16C)

Specify the digital filter width of the digital debounce facility assigned to _NMI/_SD. Specifiable only when port P85 is used.

Specify the digital filter width of the digital debounce facility assigned to INPC17/_INT5. Specifiable only when port P17 is used.

When using any other port, set 0 for this data.

(R8C) (H8/300H)

Specify 0.

Return value	If the I/O port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	I/O port
Reference	__ReadIOPort , __WriteIOPort , __SetIOPortRegister , __ReadIOPortRegister , __WriteIOPortRegister
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable I/O ports differ with each CPU used. • The API cannot set function that the specified I/O port dont have.

Program example

```
#include " rapi_io_port_r8c_13.h

void func( void )
{
    /* Set up port P03 as input port */
    __SetIOPort(RAPI_PORT_0_3| RAPI_PORT_INPUT| RAPI_PULLED_HIGH, 0, 0 );
}
```

__ReadIOPort

Synopsis

<Read from I/O port>

Boolean __ReadIOPort(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the variable in which the value read from I/O port is stored.

Description

Gets the value of a specified I/O port.

[data1]

Specify an I/O port from which data is read. The definition values corresponding to each I/O port are listed below.

(M16C)

RAPI_PORT_0_0	Port P0 ₀	RAPI_PORT_0_1	Port P0 ₁
RAPI_PORT_0_2	Port P0 ₂	RAPI_PORT_0_3	Port P0 ₃
RAPI_PORT_0_4	Port P0 ₄	RAPI_PORT_0_5	Port P0 ₅
RAPI_PORT_0_6	Port P0 ₆	RAPI_PORT_0_7	Port P0 ₇
RAPI_PORT_1_0	Port P1 ₀	RAPI_PORT_1_1	Port P1 ₁
RAPI_PORT_1_2	Port P1 ₂	RAPI_PORT_1_3	Port P1 ₃
RAPI_PORT_1_4	Port P1 ₄	RAPI_PORT_1_5	Port P1 ₅
RAPI_PORT_1_6	Port P1 ₆	RAPI_PORT_1_7	Port P1 ₇
RAPI_PORT_2_0	Port P2 ₀	RAPI_PORT_2_1	Port P2 ₁
RAPI_PORT_2_2	Port P2 ₂	RAPI_PORT_2_3	Port P2 ₃
RAPI_PORT_2_4	Port P2 ₄	RAPI_PORT_2_5	Port P2 ₅
RAPI_PORT_2_6	Port P2 ₆	RAPI_PORT_2_7	Port P2 ₇
RAPI_PORT_3_0	Port P3 ₀	RAPI_PORT_3_1	Port P3 ₁
RAPI_PORT_3_2	Port P3 ₂	RAPI_PORT_3_3	Port P3 ₃
RAPI_PORT_3_4	Port P3 ₄	RAPI_PORT_3_5	Port P3 ₅
RAPI_PORT_3_6	Port P3 ₆	RAPI_PORT_3_7	Port P3 ₇
RAPI_PORT_6_0	Port P6 ₀	RAPI_PORT_6_1	Port P6 ₁
RAPI_PORT_6_2	Port P6 ₂	RAPI_PORT_6_3	Port P6 ₃
RAPI_PORT_6_4	Port P6 ₄	RAPI_PORT_6_5	Port P6 ₅
RAPI_PORT_6_6	Port P6 ₆	RAPI_PORT_6_7	Port P6 ₇
RAPI_PORT_7_0	Port P7 ₀	RAPI_PORT_7_1	Port P7 ₁
RAPI_PORT_7_2	Port P7 ₂	RAPI_PORT_7_3	Port P7 ₃
RAPI_PORT_7_4	Port P7 ₄	RAPI_PORT_7_5	Port P7 ₅
RAPI_PORT_7_6	Port P7 ₆	RAPI_PORT_7_7	Port P7 ₇
RAPI_PORT_8_0	Port P8 ₀	RAPI_PORT_8_1	Port P8 ₁
RAPI_PORT_8_2	Port P8 ₂	RAPI_PORT_8_3	Port P8 ₃
RAPI_PORT_8_4	Port P8 ₄	RAPI_PORT_8_5	Port P8 ₅
RAPI_PORT_8_6	Port P8 ₆	RAPI_PORT_8_7	Port P8 ₇
RAPI_PORT_9_0	Port P9 ₀	RAPI_PORT_9_1	Port P9 ₁
RAPI_PORT_9_2	Port P9 ₂	RAPI_PORT_9_3	Port P9 ₃

RAPI_PORT_9_5	Port P9 ₅	RAPI_PORT_9_6	Port P9 ₆
RAPI_PORT_9_7	Port P9 ₇	RAPI_PORT_10_0	Port P10 ₀
RAPI_PORT_10_1	Port P10 ₁	RAPI_PORT_10_2	Port P10 ₂
RAPI_PORT_10_3	Port P10 ₃	RAPI_PORT_10_4	Port P10 ₄
RAPI_PORT_10_5	Port P10 ₅	RAPI_PORT_10_6	Port P10 ₆
RAPI_PORT_10_7	Port P10 ₇		

(R8C)

RAPI_PORT_0_0	Port P0 ₀	RAPI_PORT_0_1	Port P0 ₁
RAPI_PORT_0_2	Port P0 ₂	RAPI_PORT_0_3	Port P0 ₃
RAPI_PORT_0_4	Port P0 ₄	RAPI_PORT_0_5	Port P0 ₅
RAPI_PORT_0_6	Port P0 ₆	RAPI_PORT_0_7	Port P0 ₇
RAPI_PORT_1_0	Port P1 ₀	RAPI_PORT_1_1	Port P1 ₁
RAPI_PORT_1_2	Port P1 ₂	RAPI_PORT_1_3	Port P1 ₃
RAPI_PORT_1_4	Port P1 ₄	RAPI_PORT_1_5	Port P1 ₅
RAPI_PORT_1_6	Port P1 ₆	RAPI_PORT_1_7	Port P1 ₇
RAPI_PORT_2_0	Port P2 ₀	RAPI_PORT_2_1	Port P2 ₁
RAPI_PORT_2_2	Port P2 ₂	RAPI_PORT_2_3	Port P2 ₃
RAPI_PORT_2_4	Port P2 ₄	RAPI_PORT_2_5	Port P2 ₅
RAPI_PORT_2_6	Port P2 ₆	RAPI_PORT_2_7	Port P2 ₇
RAPI_PORT_3_0	Port P3 ₀	RAPI_PORT_3_1	Port P3 ₁
RAPI_PORT_3_2	Port P3 ₂	RAPI_PORT_3_3	Port P3 ₃
RAPI_PORT_3_4	Port P3 ₄	RAPI_PORT_3_5	Port P3 ₅
RAPI_PORT_3_7	Port P3 ₇	RAPI_PORT_4_2	Port P4 ₂
RAPI_PORT_4_3	Port P4 ₃	RAPI_PORT_4_4	Port P4 ₄
RAPI_PORT_4_5	Port P4 ₅	RAPI_PORT_4_6	Port P4 ₆
RAPI_PORT_4_7	Port P4 ₇	RAPI_PORT_6_0	Port P6 ₀
RAPI_PORT_6_1	Port P6 ₁	RAPI_PORT_6_2	Port P6 ₂
RAPI_PORT_6_3	Port P6 ₃	RAPI_PORT_6_4	Port P6 ₄
RAPI_PORT_6_5	Port P6 ₅	RAPI_PORT_6_6	Port P6 ₆
RAPI_PORT_6_7	Port P6 ₇		

(H8/300H)

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_4	Port P14
RAPI_PORT_1_5	Port P15	RAPI_PORT_1_6	Port P16
RAPI_PORT_1_7	Port P17	RAPI_PORT_2_0	Port P20
RAPI_PORT_2_1	Port P21	RAPI_PORT_2_2	Port P22
RAPI_PORT_2_3	Port P23	RAPI_PORT_2_4	Port P24
RAPI_PORT_2_5	Port P25	RAPI_PORT_2_6	Port P26
RAPI_PORT_2_7	Port P27	RAPI_PORT_3_0	Port P30
RAPI_PORT_3_1	Port P31	RAPI_PORT_3_2	Port P32
RAPI_PORT_3_3	Port P33	RAPI_PORT_3_4	Port P34
RAPI_PORT_3_5	Port P35	RAPI_PORT_3_6	Port P36
RAPI_PORT_3_7	Port P37	RAPI_PORT_5_0	Port P50

RAPI_PORT_5_1	PortP51	RAPI_PORT_5_2	PortP52
RAPI_PORT_5_3	PortP53	RAPI_PORT_5_4	PortP54
RAPI_PORT_5_5	PortP55	RAPI_PORT_5_6	PortP56
RAPI_PORT_5_7	PortP57	RAPI_PORT_6_0	PortP60
RAPI_PORT_6_1	PortP61	RAPI_PORT_6_2	PortP62
RAPI_PORT_6_3	PortP63	RAPI_PORT_6_4	PortP64
RAPI_PORT_6_5	PortP65	RAPI_PORT_6_6	PortP66
RAPI_PORT_6_7	PortP67	RAPI_PORT_7_0	PortP70
RAPI_PORT_7_1	PortP71	RAPI_PORT_7_2	PortP72
RAPI_PORT_7_4	PortP74	RAPI_PORT_7_5	PortP75
RAPI_PORT_7_6	PortP76	RAPI_PORT_7_7	PortP77
RAPI_PORT_8_0	PortP80	RAPI_PORT_8_1	PortP81
RAPI_PORT_8_2	PortP82	RAPI_PORT_8_3	PortP83
RAPI_PORT_8_4	PortP84	RAPI_PORT_8_5	PortP85
RAPI_PORT_8_6	PortP86	RAPI_PORT_8_7	PortP87
RAPI_PORT_9_0	PortP90	RAPI_PORT_9_1	PortP91
RAPI_PORT_9_2	PortP92	RAPI_PORT_9_3	PortP93
RAPI_PORT_9_4	PortP94	RAPI_PORT_9_5	PortP95
RAPI_PORT_9_6	PortP96	RAPI_PORT_9_7	PortP97
RAPI_PORT_B_0	PortPB0	RAPI_PORT_B_1	PortPB1
RAPI_PORT_B_2	PortPB2	RAPI_PORT_B_3	PortPB3
RAPI_PORT_B_4	PortPB4	RAPI_PORT_B_5	PortPB5
RAPI_PORT_B_6	PortPB6	RAPI_PORT_B_7	PortPB7
RAPI_PORT_C_0	PortPC0	RAPI_PORT_C_1	PortPC1
RAPI_PORT_C_2	PortPC2	RAPI_PORT_C_3	PortPC3
RAPI_PORT_D_0	PortPD0	RAPI_PORT_D_1	PortPD1
RAPI_PORT_D_2	PortPD2	RAPI_PORT_D_3	PortPD3
RAPI_PORT_D_4	PortPD4	RAPI_PORT_D_5	PortPD5
RAPI_PORT_D_6	PortPD6	RAPI_PORT_D_7	PortPD7
RAPI_PORT_E_0	PortPE0	RAPI_PORT_E_1	PortPE1
RAPI_PORT_E_2	PortPE2	RAPI_PORT_E_3	PortPE3
RAPI_PORT_E_4	PortPE4	RAPI_PORT_E_5	PortPE5
RAPI_PORT_E_6	PortPE6	RAPI_PORT_E_7	PortPE7
RAPI_PORT_F_0	PortPF0	RAPI_PORT_F_1	PortPF1
RAPI_PORT_F_2	PortPF2	RAPI_PORT_F_3	PortPF3
RAPI_PORT_F_4	PortPF4	RAPI_PORT_F_5	PortPF5
RAPI_PORT_F_6	PortPF6	RAPI_PORT_F_7	PortPF7
RAPI_PORT_G_0	PortPG0	RAPI_PORT_G_1	PortPG1
RAPI_PORT_G_2	PortPG2	RAPI_PORT_G_3	PortPG3
RAPI_PORT_G_4	PortPG4	RAPI_PORT_G_5	PortPG5
RAPI_PORT_G_6	PortPG6	RAPI_PORT_G_7	PortPG7
RAPI_PORT_H_0	PortPH0	RAPI_PORT_H_1	PortPH1
RAPI_PORT_H_2	PortPH2	RAPI_PORT_H_3	PortPH3

RAPI_PORT_H_4	PortPH4	RAPI_PORT_H_5	PortPH5
RAPI_PORT_H_6	PortPH6	RAPI_PORT_H_7	PortPH7
RAPI_PORT_J_0	PortPJ0	RAPI_PORT_J_1	PortPJ1

Return value

If the I/O port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

[__SetIOPort](#), [__WriteIOPort](#), [__SetIOPortRegister](#), [__ReadIOPortRegister](#), [__WriteIOPortRegister](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O ports differ with each CPU used.
- Ports in port B of the H8/300H that are used as analog input pins cannot be used as input ports.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    /* Get the value of port P12 */
    __ReadIOPort(RAPI_PORT_1_2, &data );
}
```

__WritelOPort

Synopsis

<Write to I/O port>

Boolean __WritelOPort(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Data to be written to I/O port

Description

Writes data to a specified I/O port.

[data1]

Specify an I/O port to which data is written. The definition values corresponding to each I/O port are listed below.

(M16C)

RAPI_PORT_0_0	Port P0 ₀	RAPI_PORT_0_1	Port P0 ₁
RAPI_PORT_0_2	Port P0 ₂	RAPI_PORT_0_3	Port P0 ₃
RAPI_PORT_0_4	Port P0 ₄	RAPI_PORT_0_5	Port P0 ₅
RAPI_PORT_0_6	Port P0 ₆	RAPI_PORT_0_7	Port P0 ₇
RAPI_PORT_1_0	Port P1 ₀	RAPI_PORT_1_1	Port P1 ₁
RAPI_PORT_1_2	Port P1 ₂	RAPI_PORT_1_3	Port P1 ₃
RAPI_PORT_1_4	Port P1 ₄	RAPI_PORT_1_5	Port P1 ₅
RAPI_PORT_1_6	Port P1 ₆	RAPI_PORT_1_7	Port P1 ₇
RAPI_PORT_2_0	Port P2 ₀	RAPI_PORT_2_1	Port P2 ₁
RAPI_PORT_2_2	Port P2 ₂	RAPI_PORT_2_3	Port P2 ₃
RAPI_PORT_2_4	Port P2 ₄	RAPI_PORT_2_5	Port P2 ₅
RAPI_PORT_2_6	Port P2 ₆	RAPI_PORT_2_7	Port P2 ₇
RAPI_PORT_3_0	Port P3 ₀	RAPI_PORT_3_1	Port P3 ₁
RAPI_PORT_3_2	Port P3 ₂	RAPI_PORT_3_3	Port P3 ₃
RAPI_PORT_3_4	Port P3 ₄	RAPI_PORT_3_5	Port P3 ₅
RAPI_PORT_3_6	Port P3 ₆	RAPI_PORT_3_7	Port P3 ₇
RAPI_PORT_6_0	Port P6 ₀	RAPI_PORT_6_1	Port P6 ₁
RAPI_PORT_6_2	Port P6 ₂	RAPI_PORT_6_3	Port P6 ₃
RAPI_PORT_6_4	Port P6 ₄	RAPI_PORT_6_5	Port P6 ₅
RAPI_PORT_6_6	Port P6 ₆	RAPI_PORT_6_7	Port P6 ₇
RAPI_PORT_7_0	Port P7 ₀	RAPI_PORT_7_1	Port P7 ₁
RAPI_PORT_7_2	Port P7 ₂	RAPI_PORT_7_3	Port P7 ₃
RAPI_PORT_7_4	Port P7 ₄	RAPI_PORT_7_5	Port P7 ₅
RAPI_PORT_7_6	Port P7 ₆	RAPI_PORT_7_7	Port P7 ₇
RAPI_PORT_8_0	Port P8 ₀	RAPI_PORT_8_1	Port P8 ₁
RAPI_PORT_8_2	Port P8 ₂	RAPI_PORT_8_3	Port P8 ₃
RAPI_PORT_8_4	Port P8 ₄	RAPI_PORT_8_5	Port P8 ₅
RAPI_PORT_8_6	Port P8 ₆	RAPI_PORT_8_7	Port P8 ₇
RAPI_PORT_9_0	Port P9 ₀	RAPI_PORT_9_1	Port P9 ₁
RAPI_PORT_9_2	Port P9 ₂	RAPI_PORT_9_3	Port P9 ₃

RAPI_PORT_9_5	Port P9 ₅	RAPI_PORT_9_6	Port P9 ₆
RAPI_PORT_9_7	Port P9 ₇	RAPI_PORT_10_0	Port P10 ₀
RAPI_PORT_10_1	Port P10 ₁	RAPI_PORT_10_2	Port P10 ₂
RAPI_PORT_10_3	Port P10 ₃	RAPI_PORT_10_4	Port P10 ₄
RAPI_PORT_10_5	Port P10 ₅	RAPI_PORT_10_6	Port P10 ₆
RAPI_PORT_10_7	Port P10 ₇		

(R8C)

RAPI_PORT_0_0	Port P0 ₀	RAPI_PORT_0_1	Port P0 ₁
RAPI_PORT_0_2	Port P0 ₂	RAPI_PORT_0_3	Port P0 ₃
RAPI_PORT_0_4	Port P0 ₄	RAPI_PORT_0_5	Port P0 ₅
RAPI_PORT_0_6	Port P0 ₆	RAPI_PORT_0_7	Port P0 ₇
RAPI_PORT_1_0	Port P1 ₀	RAPI_PORT_1_1	Port P1 ₁
RAPI_PORT_1_2	Port P1 ₂	RAPI_PORT_1_3	Port P1 ₃
RAPI_PORT_1_4	Port P1 ₄	RAPI_PORT_1_5	Port P1 ₅
RAPI_PORT_1_6	Port P1 ₆	RAPI_PORT_1_7	Port P1 ₇
RAPI_PORT_2_0	Port P2 ₀	RAPI_PORT_2_1	Port P2 ₁
RAPI_PORT_2_2	Port P2 ₂	RAPI_PORT_2_3	Port P2 ₃
RAPI_PORT_2_4	Port P2 ₄	RAPI_PORT_2_5	Port P2 ₅
RAPI_PORT_2_6	Port P2 ₆	RAPI_PORT_2_7	Port P2 ₇
RAPI_PORT_3_0	Port P3 ₀	RAPI_PORT_3_1	Port P3 ₁
RAPI_PORT_3_2	Port P3 ₂	RAPI_PORT_3_3	Port P3 ₃
RAPI_PORT_3_4	Port P3 ₄	RAPI_PORT_3_5	Port P3 ₅
RAPI_PORT_3_7	Port P3 ₇	RAPI_PORT_4_3	Port P4 ₃
RAPI_PORT_4_4	Port P4 ₄	RAPI_PORT_4_5	Port P4 ₅
RAPI_PORT_6_0	Port P6 ₀	RAPI_PORT_6_1	Port P6 ₁
RAPI_PORT_6_2	Port P6 ₂	RAPI_PORT_6_3	Port P6 ₃
RAPI_PORT_6_4	Port P6 ₄	RAPI_PORT_6_5	Port P6 ₅
RAPI_PORT_6_6	Port P6 ₆	RAPI_PORT_6_7	Port P6 ₇

(H8/300H)

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_4	Port P14
RAPI_PORT_1_5	Port P15	RAPI_PORT_1_6	Port P16
RAPI_PORT_1_7	Port P17	RAPI_PORT_2_0	Port P20
RAPI_PORT_2_1	Port P21	RAPI_PORT_2_2	Port P22
RAPI_PORT_2_3	Port P23	RAPI_PORT_2_4	Port P24
RAPI_PORT_2_5	Port P25	RAPI_PORT_2_6	Port P26
RAPI_PORT_2_7	Port P27	RAPI_PORT_3_0	Port P30
RAPI_PORT_3_1	Port P31	RAPI_PORT_3_2	Port P32
RAPI_PORT_3_3	Port P33	RAPI_PORT_3_4	Port P34
RAPI_PORT_3_5	Port P35	RAPI_PORT_3_6	Port P36
RAPI_PORT_3_7	Port P37	RAPI_PORT_5_0	Port P50
RAPI_PORT_5_1	Port P51	RAPI_PORT_5_2	Port P52
RAPI_PORT_5_3	Port P53	RAPI_PORT_5_4	Port P54

RAPI_PORT_5_5	PortP55	RAPI_PORT_5_6	PortP56
RAPI_PORT_5_7	PortP57	RAPI_PORT_6_0	PortP60
RAPI_PORT_6_1	PortP61	RAPI_PORT_6_2	PortP62
RAPI_PORT_6_3	PortP63	RAPI_PORT_6_4	PortP64
RAPI_PORT_6_5	PortP65	RAPI_PORT_6_6	PortP66
RAPI_PORT_6_7	PortP67	RAPI_PORT_7_0	PortP70
RAPI_PORT_7_1	PortP71	RAPI_PORT_7_2	PortP72
RAPI_PORT_7_4	PortP74	RAPI_PORT_7_5	PortP75
RAPI_PORT_7_6	PortP76	RAPI_PORT_7_7	PortP77
RAPI_PORT_8_0	PortP80	RAPI_PORT_8_1	PortP81
RAPI_PORT_8_2	PortP82	RAPI_PORT_8_3	PortP83
RAPI_PORT_8_4	PortP84	RAPI_PORT_8_5	PortP85
RAPI_PORT_8_6	PortP86	RAPI_PORT_8_7	PortP87
RAPI_PORT_9_0	PortP90	RAPI_PORT_9_1	PortP91
RAPI_PORT_9_2	PortP92	RAPI_PORT_9_3	PortP93
RAPI_PORT_9_4	PortP94	RAPI_PORT_9_5	PortP95
RAPI_PORT_9_6	PortP96	RAPI_PORT_9_7	PortP97
RAPI_PORT_C_0	PortPC0	RAPI_PORT_C_1	PortPC1
RAPI_PORT_C_2	PortPC2	RAPI_PORT_C_3	PortPC3
RAPI_PORT_D_0	PortPD0	RAPI_PORT_D_1	PortPD1
RAPI_PORT_D_2	PortPD2	RAPI_PORT_D_3	PortPD3
RAPI_PORT_D_4	PortPD4	RAPI_PORT_D_5	PortPD5
RAPI_PORT_D_6	PortPD6	RAPI_PORT_D_7	PortPD7
RAPI_PORT_E_0	PortPE0	RAPI_PORT_E_1	PortPE1
RAPI_PORT_E_2	PortPE2	RAPI_PORT_E_3	PortPE3
RAPI_PORT_E_4	PortPE4	RAPI_PORT_E_5	PortPE5
RAPI_PORT_E_6	PortPE6	RAPI_PORT_E_7	PortPE7
RAPI_PORT_G_0	PortPG0	RAPI_PORT_G_1	PortPG1
RAPI_PORT_G_2	PortPG2	RAPI_PORT_G_3	PortPG3
RAPI_PORT_G_4	PortPG4	RAPI_PORT_G_5	PortPG5
RAPI_PORT_G_6	PortPG6	RAPI_PORT_G_7	PortPG7
RAPI_PORT_H_0	PortPH0	RAPI_PORT_H_1	PortPH1
RAPI_PORT_H_2	PortPH2	RAPI_PORT_H_3	PortPH3
RAPI_PORT_H_4	PortPH4	RAPI_PORT_H_5	PortPH5
RAPI_PORT_H_6	PortPH6	RAPI_PORT_H_7	PortPH7
RAPI_PORT_J_0	PortPJ0	RAPI_PORT_J_1	PortPJ1

Return value

If the I/O port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

[__SetIOPort](#), [__ReadIOPort](#), [__SetIOPortRegister](#), [__ReadIOPortRegister](#),

[__WriteIOPortRegister](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O ports differ with each CPU used.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    unsigned int data;

    /* Set the data to port P05 */
    __WriteIOPort( RAPI_PORT_0_5, 0 );
}
```

__SetIOPortRegister

Synopsis

<Set I/O port register>

Boolean __SetIOPortRegister(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)

Description

[data1]

Set the operating condition of a specified I/O port in each relevant register.

(M16C)

The definition values corresponding to each I/O port register are listed below.

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_10	Port P10 register		

The definition values related to port settings are described below.

RAPI_LATCH	Set to read the port latch regardless of whether the port is set for input or output. Specifiable only when port P1 is used.
------------	--

(R8C)

The definition values corresponding to each I/O port register are listed below.

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_4	Port P4 register	RAPI_PORT_6	Port P6 register

(H8/300H)

The definition values corresponding to each I/O port register are listed below.

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_C	Port PC register	RAPI_PORT_D	Port PD register
RAPI_PORT_E	Port PE register	RAPI_PORT_F	Port PF register
RAPI_PORT_G	Port PG register	RAPI_PORT_H	Port PH register
RAPI_PORT_J	Port PJ register		

[data2]

(M16C) (R8C)

Specify the set value for the port direction register corresponding to a selected port.

(H8/300H)

Specify the set value for the port control register corresponding to a selected port.

[data3]

(M16C) (R8C)

Specify the set value for the pullup control register corresponding to a selected port.
(H8/300H)

Specify the set value for the pullup control register corresponding to a selected port.

[data4]

(M16C)

Specify the digital filter width of the digital debounce facility assigned to `_NMI/_SD`.
Specifiable only when port P8 is used.

Specify the digital filter width of the digital debounce facility assigned to `INPC17/_INT5`. Specifiable only when port P1 is used.

When using any other port, set 0 for this data.

(R8C)

Specify the set value for the port P1 drive capacity register. Specifiable only when port P1 is used.

When using any other port, set 0 for this data.

(H8/300H)

Specify 0.

Return value

If the I/O port register specification is incorrect, `RAPI_FALSE` is returned; otherwise, `RAPI_TRUE` is returned.

Functionality

I/O port

Reference

[__SetIOPort](#), [__ReadIOPort](#), [__WriteIOPort](#), [__ReadIOPortRegister](#),
[__WriteIOPortRegister](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O port registers differ with each CPU used.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    /* Set inputs/outputs of port P1 register */
    __SetIOPortRegister(RAPI_PORT_1, 0xAA, 0, 0 );
}
```

__ReadIOPortRegister

Synopsis

<Read from I/O port register>

Boolean __ReadIOPortRegister(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the variable in which the value read from I/O port register is stored.

Description

Gets the value of a specified I/O port from each relevant register.

[data1]

Specify an I/O port register from which data is read. The definition values corresponding to each I/O port register are listed below.

(M16C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_10	Port P10 register		

(R8C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_4	Port P4 register	RAPI_PORT_6	Port P6 register

(H8/300H)

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_B	Port PB register	RAPI_PORT_C	Port PC register
RAPI_PORT_D	Port PD register	RAPI_PORT_E	Port PE register
RAPI_PORT_F	Port PF register	RAPI_PORT_G	Port PG register
RAPI_PORT_H	Port PH register	RAPI_PORT_J	Port PJ register

Return value

If the I/O port register specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

[__SetIOPort](#), [__ReadIOPort](#), [__WriteIOPort](#), [__SetIOPortRegister](#),
[__WriteIOPortRegister](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O port registers differ with each CPU used.

- Ports in port B of the H8/300H that are used as analog input pins cannot be used as input ports.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    unsigned int data;

    /* Get the value of port P1 register */
    __ReadIOPortRegister( RAPI_PORT_1, &data );
}
```

__WritelPortRegister

Synopsis

<Write to I/O port register>

Boolean __WritelPortRegister(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Data to be written to I/O port register

Description

Writes the value for a specified I/O port to each relevant register.

[data1]

Specify an I/O port register to which data is written. The definition values corresponding to each I/O port register are listed below.

(M16C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_10	Port P10 register		

(R8C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_4	Port P4 register	RAPI_PORT_6	Port P6 register

(H8/300H)

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_B	Port PB register	RAPI_PORT_C	Port PC register
RAPI_PORT_D	Port PD register	RAPI_PORT_E	Port PE register
RAPI_PORT_F	Port PF register	RAPI_PORT_G	Port PG register
RAPI_PORT_H	Port PH register	RAPI_PORT_J	Port PJ register

Return value

If the I/O port register specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

[__SetIOPort](#), [__ReadIOPort](#), [__WritelPort](#), [__SetIOPortRegister](#),
[__ReadIOPortRegister](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O port registers differ with each CPU used.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    /* Set the data to port P1 register */
    __WriteIOPortRegister( RAPI_PORT_1, 0xFF );
}
```

4.2.4 External interrupt

__SetInterrupt

Synopsis

<Set external interrupt>

Boolean **__SetInterrupt(unsigned long data1, unsigned int data2, void* func)**

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified external interrupt.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value. Note, however, that multiple external interrupts cannot be specified at the same time.

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.
RAPI_INT_RISING	Specifies a rising edge for the active edge of a selected external interrupt.
RAPI_INT_FALLING	Specifies a falling edge for the active edge of a selected external interrupt.
RAPI_INT_BOTH	Specifies both edges for the active edge of a selected external interrupt.
RAPI_KI0_ENABLE	Uses _KI0 pin input.
RAPI_KI1_ENABLE	Uses _KI1 pin input.
RAPI_KI2_ENABLE	Uses _KI2 pin input.
RAPI_KI3_ENABLE	Uses _KI3 pin input.

- **Specifiable definition values when _INT0-5 interrupts are used (RAPI_INT0 to RAPI_INT5 specified)**

(Polarity) Specify one from { RAPI_INT_RISING, RAPI_INT_FALLING, RAPI_INT_BOTH }. The default value is RAPI_INT_FALLING.

- **Specifiable definition values when key input interrupt is used (RAPI_KEY specified)**

(Input pin) To use _KI0, _KI1, _KI2, or _KI3 pin input, specify RAPI_KI0_ENABLE, RAPI_KI1_ENABLE, RAPI_KI2_ENABLE, or RAPI_KI3_ENABLE, respectively. The default value is RAPI_INT_FALLING.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.

RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.
RAPI_INT_RISING	Specifies a rising edge for the active edge of a selected external interrupt.
RAPI_INT_FALLING	Specifies a falling edge for the active edge of a selected external interrupt.
RAPI_INT_BOTH	Specifies both edges for the active edge of a selected external interrupt.
RAPI_FILTER_F1	Uses the digital filter facility that has a sampling frequency f_1 .
RAPI_FILTER_F8	Uses the digital filter facility that has a sampling frequency f_8 .
RAPI_FILTER_F32	Uses the digital filter facility that has a sampling frequency f_{32} .
RAPI_KI0_FALLING	Enables _KI0 pin input whose falling edge is the active edge.
RAPI_KI0_RISING	Enables _KI0 pin input whose rising edge is the active edge.
RAPI_KI1_FALLING	Enables _KI1 pin input whose falling edge is the active edge.
RAPI_KI1_RISING	Enables _KI1 pin input whose rising edge is the active edge.
RAPI_KI2_FALLING	Enables _KI2 pin input whose falling edge is the active edge.
RAPI_KI2_RISING	Enables _KI2 pin input whose rising edge is the active edge.
RAPI_KI3_FALLING	Enables _KI3 pin input whose falling edge is the active edge.
RAPI_KI3_RISING	Enables _KI3 pin input whose rising edge is the active edge.

• **Specifiable definition values when _INT0 to _INT3 interrupt is used (RAPI_INT0 to RAPI_INT3 specified)**

(Polarity) Specify one from { RAPI_INT_RISING, RAPI_INT_FALLING, RAPI_INT_BOTH }. The default value is RAPI_INT_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32 }. If no filters are specified, "No filter" is set.

• **Specifiable definition values when key input interrupt is used (RAPI_KEY specified)**

(_KI0 pin input) Specify one from { RAPI_KI0_FALLING, RAPI_KI0_RISING }. If _KI0 pin input is not specified, "_KI0 pin input disabled" is set.

(_KI1 pin input) Specify one from { RAPI_KI1_FALLING, RAPI_KI1_RISING }. If _KI1 pin input is not specified, "_KI1 pin input disabled" is set.

(_KI2 pin input) Specify one from { RAPI_KI2_FALLING, RAPI_KI2_RISING }. If _KI2 pin input is not specified, "_KI2 pin input disabled" is set.

(_KI3 pin input) Specify one from { RAPI_KI3_FALLING, RAPI_KI3_RISING }. If _KI3 pin input is not specified, "_KI3 pin input disabled" is set.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP	Uses WKP interrupt.
RAPI_NOT_INT_REQUEST	Does not request assertion of interrupt for a selected external interrupt.
RAPI_INT_REQUEST	Requests assertion of interrupt for a selected external interrupt.

RAPI_INT_RISING	Specifies a rising edge for the active edge of a selected external interrupt.
RAPI_INT_FALLING	Specifies a falling edge for the active edge of a selected external interrupt.
RAPI_WKP0_FALLING	Enables _WKP0 pin input whose falling edge is the active edge.
RAPI_WKP0_RISING	Enables _WKP0 pin input whose rising edge is the active edge.
RAPI_WKP1_FALLING	Enables _WKP1 pin input whose falling edge is the active edge.
RAPI_WKP1_RISING	Enables _WKP1 pin input whose rising edge is the active edge.
RAPI_WKP2_FALLING	Enables _WKP2 pin input whose falling edge is the active edge.
RAPI_WKP2_RISING	Enables _WKP2 pin input whose rising edge is the active edge.
RAPI_WKP3_FALLING	Enables _WKP3 pin input whose falling edge is the active edge.
RAPI_WKP3_RISING	Enables _WKP3 pin input whose rising edge is the active edge.
RAPI_WKP4_FALLING	Enables _WKP4 pin input whose falling edge is the active edge.
RAPI_WKP4_RISING	Enables _WKP4 pin input whose rising edge is the active edge.
RAPI_WKP5_FALLING	Enables _WKP5 pin input whose falling edge is the active edge.
RAPI_WKP5_RISING	Enables _WKP5 pin input whose rising edge is the active edge.

• **Specifiable definition values when _IRQ0–3 interrupts are used (RAPI_IRQ0 to RAPI_IRQ3 specified)**

(Polarity) Specify one from { RAPI_INT_RISING, RAPI_INT_FALLING }. The default value is RAPI_INT_FALLING.

(Interrupt request) Specify one from { RAPI_NOT_INT_REQUEST, RAPI_INT_REQUEST }. The default value is RAPI_NOT_INT_REQUEST.

• **Specifiable definition values when WKP interrupt is used (RAPI_WKP specified)**

(_WKP0 pin input) Specify one from { RAPI_WKP0_FALLING, RAPI_WKP0_RISING }. If _WKP0 pin input is not specified, “_WKP0 pin input disabled” is set.

(_WKP1 pin input) Specify one from { RAPI_WKP1_FALLING, RAPI_WKP1_RISING }. If _WKP1 pin input is not specified, “_WKP1 pin input disabled” is set.

(_WKP2 pin input) Specify one from { RAPI_WKP2_FALLING, RAPI_WKP2_RISING }. If _WKP2 pin input is not specified, “_WKP2 pin input disabled” is set.

(_WKP3 pin input) Specify one from { RAPI_WKP3_FALLING, RAPI_WKP3_RISING }. If _WKP3 pin input is not specified, “_WKP3 pin input disabled” is set.

(_WKP4 pin input) Specify one from { RAPI_WKP4_FALLING, RAPI_WKP4_RISING }. If _WKP4 pin input is not specified, “_WKP4 pin input disabled” is set.

(_WKP5 pin input) Specify one from { RAPI_WKP5_FALLING, RAPI_WKP5_RISING }. If _WKP5 pin input is not specified, “_WKP5 pin input disabled” is set.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

Return value

If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise,

RAPI_TRUE is returned.

Functionality

External interrupt

Reference

[__EnableInterrupt](#), [__GetInterruptFlag](#), [__ClearInterruptFlag](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable external interrupts differ with each CPU used.
- For the R8C, if its CPU type is the one that does not accept specification of “both edges” for interrupts to INT2 and INT3, RAPI_INT_BOTH cannot be specified for INT2 and INT3.

Program example

```
#include " rapi_interrupt_r8c_13.h"

void IntFunc( void ){}

void func( void )
{
    /* Set up _INT0 interrupt */
    __SetInterrupt( RAPI_INT0|RAPI_INT_FALLING, 0, IntFunc );
}
```

__EnableInterrupt

Synopsis

<Control external interrupt>

Boolean __EnableInterrupt(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)

Description

Changes the operating condition of a specified external interrupt.

[data1]

The following definition values can be set for data1. To specify multiple definition values at the same time, use the symbol “|” to separate each specified value. Note, however, that multiple external interrupts cannot be specified at the same time.

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP	Uses WKP interrupt.
RAPI_NOT_INT_REQUEST	Does not request assertion of interrupt for a selected external interrupt. (Default)
RAPI_INT_REQUEST	Requests assertion of interrupt for a selected external interrupt.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

Return value If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality External interrupt

Reference [__SetInterrupt](#), [__GetInterruptFlag](#), [__ClearInterruptFlag](#)

- Remark**
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
 - The specifiable external interrupts differ with each CPU used.

Program example

```
#include " rapi_interrupt_r8c_13.h"

void func( void )
{
    /* Activate _INT1 interrupt ( interrupt priority level 5 ) */
    __EnableInterrupt(RAPI_INT1, 5 );
}
```

__GetInterruptFlag

Synopsis

<Get the status of external interrupt flag>

Boolean __GetInterruptFlag(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which the acquired flag data is stored

Description

Gets the value of interrupt request flag of a specified external interrupt.

[data1]

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP0	Uses WKP0 interrupt.
RAPI_WKP1	Uses WKP1 interrupt.
RAPI_WKP2	Uses WKP2 interrupt.
RAPI_WKP3	Uses WKP3 interrupt.
RAPI_WKP4	Uses WKP4 interrupt.
RAPI_WKP5	Uses WKP5 interrupt.

[data2]

0	0	0	0	0	0	0	0	/
---	---	---	---	---	---	---	---	---

Value of interrupt request flag (0: Interrupt not requested; 1: Interrupt requested)

Return value

If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality	External interrupt
Reference	__SetInterrupt , __EnableInterrupt , __ClearInterruptFlag
Remark	<ul style="list-style-type: none"> • If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable external interrupts differ with each CPU used.

Program example	<pre> #include " rapi_interrupt_r8c_13.h" void func(void) { unsigned char data; /* Get flag of _INT2 interrupt */ __GetInterruptFlag(IS_INT2, &data); } </pre>
------------------------	--

__ClearInterruptFlag

Synopsis

<Clear external interrupt flag>

Boolean __ClearInterruptFlag(unsigned long data)

data	Setup data (content differs with MCU type)
------	--

Description

Clears the interrupt request flag of a specified external interrupt.

[data]

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP0	Uses WKP0 interrupt.
RAPI_WKP1	Uses WKP1 interrupt.
RAPI_WKP2	Uses WKP2 interrupt.
RAPI_WKP3	Uses WKP3 interrupt.
RAPI_WKP4	Uses WKP4 interrupt.
RAPI_WKP5	Uses WKP5 interrupt.

Return value

If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

External interrupt

Reference

[__SetInterrupt](#), [__EnableInterrupt](#), [__GetInterruptFlag](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot

be guaranteed.

- The specifiable external interrupts differ with each CPU used.

Program example

```
#include " rapi_interrupt_r8c_13.h"

void func( void )
{
    /* Clear status of _INT0 interrupt */
    __ClearInterruptFlag( RAPI_INT0 );
}
```

4.2.5 A/D converter

__CreateADC

Synopsis

<Set A/D converter>

Boolean **__CreateADC**(unsigned long data1, unsigned int data2, unsigned int data3, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Number of analog input pins used by A/D converter (differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets the A/D converter to specified mode and operating condition.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value. Note, however, that multiple analog input pin symbols cannot be specified at the same time.

(M16C)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_SINGLE_SWEEP	Selects single sweep mode.
RAPI_REPEAT_SWEEP0	Selects repeat sweep mode 0.
RAPI_REPEAT_SWEEP1	Selects repeat sweep mode 1.
RAPI_SIMULTANEOUS_SAMPLE_SWEEP	Selects simultaneous sampling sweep mode.
RAPI_DELAYED_TRIGGER0	Selects delayed trigger mode 0.
RAPI_DELAYED_TRIGGER1	Selects delayed trigger mode 1.
RAPI_AN0	Uses AN ₀ pin for the analog input pin.
RAPI_AN1	Uses AN ₁ pin for the analog input pin.
RAPI_AN2	Uses AN ₂ pin for the analog input pin.
RAPI_AN3	Uses AN ₃ pin for the analog input pin.
RAPI_AN4	Uses AN ₄ pin for the analog input pin.
RAPI_AN5	Uses AN ₅ pin for the analog input pin.
RAPI_AN6	Uses AN ₆ pin for the analog input pin.
RAPI_AN7	Uses AN ₇ pin for the analog input pin.
RAPI_AN00	Uses AN ₀₀ pin for the analog input pin.
RAPI_AN01	Uses AN ₀₁ pin for the analog input pin.
RAPI_AN02	Uses AN ₀₂ pin for the analog input pin.
RAPI_AN03	Uses AN ₀₃ pin for the analog input pin.
RAPI_AN04	Uses AN ₀₄ pin for the analog input pin.
RAPI_AN05	Uses AN ₀₅ pin for the analog input pin.
RAPI_AN06	Uses AN ₀₆ pin for the analog input pin.
RAPI_AN07	Uses AN ₀₇ pin for the analog input pin.
RAPI_AN20	Uses AN ₂₀ pin for the analog input pin.
RAPI_AN21	Uses AN ₂₁ pin for the analog input pin.

RAPI_AN22	Uses AN ₂₂ pin for the analog input pin.
RAPI_AN23	Uses AN ₂₃ pin for the analog input pin.
RAPI_AN24	Uses AN ₂₄ pin for the analog input pin.
RAPI_AN25	Uses AN ₂₅ pin for the analog input pin.
RAPI_AN26	Uses AN ₂₆ pin for the analog input pin.
RAPI_AN27	Uses AN ₂₇ pin for the analog input pin.
RAPI_AN30	Uses AN ₃₀ pin for the analog input pin.
RAPI_AN31	Uses AN ₃₁ pin for the analog input pin.
RAPI_AN32	Uses AN ₃₂ pin for the analog input pin.
RAPI_P0_GROUP	Uses port P ₀ group for the analog input pin.
RAPI_P10_GROUP	Uses port P ₁₀ group for the analog input pin.
RAPI_P1P9_GROUP	Uses port P ₁ /P ₉ group for the analog input pin.
RAPI_P9_GROUP	Uses port P ₉ group for the analog input pin.
RAPI_FAD	Sets the AD converter's operating frequency to f _{AD} .
RAPI_FAD2	Sets the AD converter's operating frequency to f _{AD} divided by 2.
RAPI_FAD3	Sets the AD converter's operating frequency to f _{AD} divided by 3.
RAPI_FAD4	Sets the AD converter's operating frequency to f _{AD} divided by 4.
RAPI_FAD6	Sets the AD converter's operating frequency to f _{AD} divided by 6.
RAPI_FAD12	Sets the AD converter's operating frequency to f _{AD} divided by 12.
RAPI_10BIT	Sets the AD converter's resolution to 10 bits.
RAPI_8BIT	Sets the AD converter's resolution to 8 bits.
RAPI_AD_ON	Sets the AD converter to start operating.
RAPI_AD_OFF	Sets the AD converter to stop operating.
RAPI_WITH_SAMPLE_HOLD	Specifies that sample-and-hold action is applied.
RAPI_WITHOUT_SAMPLE_HOLD	Specifies that sample-and-hold action is not applied.
RAPI_SOFTWARE_TRIGGER	Selects software trigger.
RAPI_EXTERNAL_TRIGGER	Selects external trigger.
RAPI_TIMER_B0_TRIGGER	Selects underflow of timer B0 as trigger.
RAPI_TIMER_B2_TRIGGER	Selects underflow of timer B2 as trigger.
RAPI_ICTB2_TRIGGER	Selects underflow of timer B2 interrupt generation frequency set counter as trigger.

• **Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)**

- (Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN00, RAPI_AN01, RAPI_AN02, RAPI_AN03, RAPI_AN04, RAPI_AN05, RAPI_AN06, RAPI_AN07, RAPI_AN20, RAPI_AN21, RAPI_AN22, RAPI_AN23, RAPI_AN24, RAPI_AN25, RAPI_AN26, RAPI_AN27, RAPI_AN30, RAPI_AN31, RAPI_AN32 }. The default is RAPI_AN0.
- (Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.
- (Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.

• **Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)**

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN00, RAPI_AN01, RAPI_AN02, RAPI_AN03, RAPI_AN04, RAPI_AN05, RAPI_AN06, RAPI_AN07, RAPI_AN20, RAPI_AN21, RAPI_AN22, RAPI_AN23, RAPI_AN24, RAPI_AN25, RAPI_AN26, RAPI_AN27, RAPI_AN30, RAPI_AN31, RAPI_AN32 }. The default is RAPI_AN0.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.

• **Specifiable definition values when single sweep mode is used (RAPI_SINGLE_SWEEP specified)**

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.

• **Specifiable definition values when repeat sweep mode 0 is used (RAPI_REPEAT_SWEEP0 specified)**

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

- (Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.
- (Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• **Specifiable definition values when repeat sweep mode 1 is used (RAPI_REPEAT_SWEEP1 specified)**

- (Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.
- (Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.
- (Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.
- (Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• **Specifiable definition values when simultaneous sampling sweep mode is used (RAPI_SIMULTANEOUS_SAMPLE_SWEEP specified)**

- (Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.
- (Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.
- (Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER, RAPI_TIMER_B0_TRIGGER, RAPI_TIMER_B2_TRIGGER, RAPI_ICTB2_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• **Specifiable definition values when delayed trigger mode 0 is used (RAPI_DELAYED_TRIGGER0 specified)**

- (Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.
- (Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Conversion method) Specify RAPI_WITH_SAMPLE_HOLD.

• **Specifiable definition values when delayed trigger mode 1 is used (RAPI_DELAYED_TRIGGER1 specified)**

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Conversion method) Specify RAPI_WITH_SAMPLE_HOLD.

(R8C)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_AN0	Uses AN0 pin for the analog input pin.
RAPI_AN1	Uses AN1 pin for the analog input pin.
RAPI_AN2	Uses AN2 pin for the analog input pin.
RAPI_AN3	Uses AN3 pin for the analog input pin.
RAPI_AN4	Uses AN4 pin for the analog input pin.
RAPI_AN5	Uses AN5 pin for the analog input pin.
RAPI_AN6	Uses AN6 pin for the analog input pin.
RAPI_AN7	Uses AN7 pin for the analog input pin.
RAPI_AN8	Uses AN8 pin for the analog input pin.
RAPI_AN9	Uses AN9 pin for the analog input pin.
RAPI_AN10	Uses AN10 pin for the analog input pin.
RAPI_AN11	Uses AN11 pin for the analog input pin.
RAPI_FAD	Sets the AD converter's operating frequency to f_{AD} .
RAPI_FAD2	Sets the AD converter's operating frequency to f_{AD} divided by 2.
RAPI_FAD4	Sets the AD converter's operating frequency to f_{AD} divided by 4.
RAPI_FOCOF	Sets the AD converter's operating frequency to fOCO-F.
RAPI_10BIT	Sets the AD converter's resolution to 10 bits.
RAPI_8BIT	Sets the AD converter's resolution to 8 bits.
RAPI_AD_ON	Sets the AD converter to start operating in __CreateAD.
RAPI_AD_OFF	Sets the AD converter to stop operating in __CreateAD
RAPI_WITH_SAMPLE_HOLD	Specifies that sample-and-hold action is applied.
RAPI_WITHOUT_SAMPLE_HOLD	Specifies that sample-and-hold action is not applied.

RAPI_SOFTWARE_TRIGGER	Selects software trigger.
RAPI_TIMER_RD_TRIGGER	Selects timer RD as trigger.

• **Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)**

- (Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11 }. Always be sure to specify an input pin.
- (Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD4, RAPI_FOCOF }. The default value is RAPI_FAD4.
- (Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.
- (Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_TIMER_RD_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• **Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)**

- (Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11}. Always be sure to specify an input pin.
- (Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD4 }. The default value is RAPI_FAD4.
- (Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.
- (Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_TIMER_RD_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

(H8/300H)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_SINGLE_SWEEP	Selects single sweep mode.
RAPI_REPEAT_SWEEP0	Selects repeat sweep mode 0.
RAPI_AN0	Uses AN0 pin for the analog input pin.
RAPI_AN1	Uses AN1 pin for the analog input pin.
RAPI_AN2	Uses AN2 pin for the analog input pin.
RAPI_AN3	Uses AN3 pin for the analog input pin.

RAPI_AN4	Uses AN4 pin for the analog input pin.
RAPI_AN5	Uses AN5 pin for the analog input pin.
RAPI_AN6	Uses AN6 pin for the analog input pin.
RAPI_AN7	Uses AN7 pin for the analog input pin.
RAPI_AN8	Uses AN8 pin for the analog input pin.
RAPI_AN9	Uses AN9 pin for the analog input pin.
RAPI_AN10	Uses AN10 pin for the analog input pin.
RAPI_AN11	Uses AN11 pin for the analog input pin.
RAPI_AN12	Uses AN12 pin for the analog input pin.
RAPI_AN13	Uses AN13 pin for the analog input pin.
RAPI_AN14	Uses AN14 pin for the analog input pin.
RAPI_AN15	Uses AN15 pin for the analog input pin.
RAPI_AN0_GROUP	Uses AN0-AN3 group for the analog input pin.
RAPI_AN4_GROUP	Uses AN4-AN7 group for the analog input pin.
RAPI_AN8_GROUP	Uses AN8-AN11 group for the analog input pin.
RAPI_AN12_GROUP	Uses AN12-AN15 group for the analog input pin.
RAPI_134_STATES	Conversion time = 134 states
RAPI_70_STATES	Conversion time = 70 states
RAPI_AD_ON	Sets the AD converter to start operating in __CreateAD.
RAPI_AD_OFF	Sets the AD converter to stop operating in __CreateAD.
RAPI_RISING_TRIGGER	Includes a rising-edge external trigger in the A/D conversion start condition.
RAPI_FALLING_TRIGGER	Includes a falling-edge external trigger in the A/D conversion start condition.
RAPI_AD_INT_REQUEST	Requests assertion of interrupt for A/D conversion interrupt.
RAPI_NOT_AD_INT_REQUEST	Does not request assertion of interrupt for A/D conversion interrupt.

• **Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)**

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11, RAPI_AN12, RAPI_AN13, RAPI_AN14, RAPI_AN15 }. The default value is RAPI_AN0.

(Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }.
If no triggers are specified, "Trigger input disabled" is set.

(Interrupt request) Specify one from { RAPI_AD_INT_REQUEST, RAPI_NOT_AD_INT_REQUEST }.
The default value is RAPI_NOT_AD_INT_REQUEST.

• **Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)**

- (Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11, RAPI_AN12, RAPI_AN13, RAPI_AN14, RAPI_AN15 } The default value is RAPI_AN0.
- (Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }.
If no triggers are specified, "Trigger input disabled" is set.

• **Specifiable definition values when single sweep mode is used (RAPI_SINGLE_SWEEP specified)**

- (Input pin) Specify one from { RAPI_AN0_GROUP, RAPI_AN4_GROUP, RAPI_AN8_GROUP, RAPI_AN12_GROUP }.
The default value is RAPI_AN0_GROUP.
- (Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }.
If no triggers are specified, "Trigger input disabled" is set.
- (Interrupt request) Specify one from { RAPI_AD_INT_REQUEST, RAPI_NOT_AD_INT_REQUEST }.
The default value is RAPI_NOT_AD_INT_REQUEST.

• **Specifiable definition values when repeat sweep mode 0 is used (RAPI_REPEAT_SWEEP0 specified)**

- (Input pin) Specify one from { RAPI_AN0_GROUP, RAPI_AN4_GROUP, RAPI_AN8_GROUP, RAPI_AN12_GROUP }. The default value is RAPI_AN0_GROUP.
- (Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.
- (Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.
- (Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }.
If no triggers are specified, "Trigger input disabled" is set.

[data2]

(M16C)

The set value differs with the A/D conversion mode used.

One-shot mode	Specify 1.
Repeat mode	
Single sweep mode	Specify 2, 4, 6, or 8.
Repeat sweep mode 0	
Simultaneous sampling mode	
Delayed trigger mode 0	

Delayed trigger mode 1	
Repeat sweep mode	Specify 1, 2, 3, or 4.

(R8C)

Specify 1.

(H8/300H)

One-shot mode	Specify 1.
Repeat mode	
Single sweep mode	Specify 1, 2, 3, or 4.
Repeat sweep mode 0	

[data3]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8/300H)

Specify the interrupt priority level (0-1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

Return value

If A/D converter was successfully set, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

[EnableADC](#), [DestroyADC](#), [GetADC](#), [GetADCAII](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable analog input pins differ with each CPU used.
- If you use an R8C MCU which does not support fOCO-F for its operating frequency, you cannot specify RAPI_FOCOF.
- If you use an R8C MCU which does not support timer RD as a trigger, you cannot specify RAPI_TIMER_RD_TRIGGER as a trigger.
- When used for the H8/300H, this API specify when freeing it from module standby.

Program example

```
#include " rapi_ad_r8c_13.h"

void AdIntFunc( void ){}

void func( void )
{
    /* Set up A/D converter as one short mode */
    __CreateADC( RAPI_ONE_SHOT|RAPI_AN2|RAPI_FAD2| RAPI_WITH_SAMPLE_HOLD
|
                RAPI_EXTERNAL_TRIGGER |RAPI_AD_ON|RAPI_10BIT, 1, 5,
AdIntFunc );
}
```

__EnableADC

Synopsis

<Control operation of A/D converter>

Boolean __EnableADC (unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Number of analog input pins used by A/D converter (differs with MCU type)

Description

Controls operation of the A/D converter by starting or stopping it.

[data1]

(M16C)

RAPI_AN0	Uses AN ₀ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN1	Uses AN ₁ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN2	Uses AN ₂ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN3	Uses AN ₃ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN4	Uses AN ₄ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN5	Uses AN ₅ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN6	Uses AN ₆ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN7	Uses AN ₇ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN00	Uses AN ₀₀ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN01	Uses AN ₀₁ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN02	Uses AN ₀₂ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN03	Uses AN ₀₃ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN04	Uses AN ₀₄ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN05	Uses AN ₀₅ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN06	Uses AN ₀₆ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN07	Uses AN ₀₇ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN20	Uses AN ₂₀ pin for the analog input pin.

	Selectable only when one-shot mode or repeat mode is used.
RAPI_AN21	Uses AN ₂₁ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN22	Uses AN ₂₂ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN23	Uses AN ₂₃ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN24	Uses AN ₂₄ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN25	Uses AN ₂₅ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN26	Uses AN ₂₆ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN27	Uses AN ₂₇ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN30	Uses AN ₃₀ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN31	Uses AN ₃₁ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN32	Uses AN ₃₂ pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_P0_GROUP	Uses port P ₀ group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_P10_GROUP	Uses port P ₁₀ group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_P1P9_GROUP	Uses port P ₁ /P ₉ group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_P9_GROUP	Uses port P ₉ group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_AD_ON	Sets the A/D converter to start operating.
RAPI_AD_OFF	Sets the A/D converter to stop operating.

(R8C)

RAPI_AN0	Uses AN ₀ pin for the analog input pin.
RAPI_AN1	Uses AN ₁ pin for the analog input pin.
RAPI_AN2	Uses AN ₂ pin for the analog input pin.
RAPI_AN3	Uses AN ₃ pin for the analog input pin.

RAPI_AN4	Uses AN ₄ pin for the analog input pin.
RAPI_AN5	Uses AN ₅ pin for the analog input pin.
RAPI_AN6	Uses AN ₆ pin for the analog input pin.
RAPI_AN7	Uses AN ₇ pin for the analog input pin.
RAPI_AN8	Uses AN ₈ pin for the analog input pin.
RAPI_AN9	Uses AN ₉ pin for the analog input pin.
RAPI_AN10	Uses AN ₁₀ pin for the analog input pin.
RAPI_AN11	Uses AN ₁₁ pin for the analog input pin.
RAPI_AD_ON	Sets the A/D converter to start operating.
RAPI_AD_OFF	Sets the A/D converter to stop operating.

(H8/300H)

RAPI_AN0	Uses AN0 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN1	Uses AN1 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN2	Uses AN2 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN3	Uses AN3 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN4	Uses AN4 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN5	Uses AN5 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN6	Uses AN6 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN7	Uses AN7 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN8	Uses AN8 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN9	Uses AN9 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN10	Uses AN10 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN11	Uses AN11 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN12	Uses AN12 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN13	Uses AN13 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN14	Uses AN14 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN15	Uses AN15 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN0_GROUP	Uses AN0-AN3 pins group for the analog input pin.

	Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AN4_GROUP	Uses AN4-AN7 pins group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AN8_GROUP	Uses AN8-AN11 pins group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AN12_GROUP	Uses AN12-AN15 pins group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AD_ON	Sets the A/D converter to start operating.
RAPI_AD_OFF	Sets the A/D converter to stop operating.

[data2]

(M16C)

The set value differs with the A/D conversion mode used.

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0 Simultaneous sampling mode Delayed trigger mode 0 Delayed trigger mode 1	Specify 2, 4, 6, or 8.
Repeat sweep mode 1	Specify 1, 2, 3, or 4.

(R8C)

Specify 1.

(H8/300H)

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0	Specify 1, 2, 3, or 4.

Return value

If A/D converter was successfully controlled, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

[__CreateADC](#), [__DestroyADC](#), [__GetADC](#), [__GetADCAI](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable analog input pins differ with each CPU used.

Program example

```
#include "rapi_ad_r8c_13.h"
```

```
void func( void )
{
    /* Disable A/D converter */
    __EnableADC( RAPI_AN0|RAPI_AD_OFF, 1 );
}
```

__DestroyADC

Synopsis

<Discard settings of A/D converter>

Boolean __DestroyADC(void)

Description

Discards settings of a specified A/D converter.

Return value

If converter setting was successfully discarded, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

[__CreateADC](#), [__EnableADC](#), [__GetADC](#), [__GetADCAI](#)

Remark

- When used for the H8/300H, this API places a specified timer into module standby state after discarding it.

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    /* Destroy A/D converter */
    __DestroyADC();
}
```

__GetADC

Synopsis

<Get A/D converted value (register specified)>

Boolean __GetADC(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which A/D converted value is stored.

Description

Gets the A/D converted value from a specified A/D register.

For data1, the following values can be set.

(M16C)

RAPI_AD0	Selects A/D register 0.
RAPI_AD1	Selects A/D register 1.
RAPI_AD2	Selects A/D register 2.
RAPI_AD3	Selects A/D register 3.
RAPI_AD4	Selects A/D register 4.
RAPI_AD5	Selects A/D register 5.
RAPI_AD6	Selects A/D register 6.
RAPI_AD7	Selects A/D register 7.

(R8C)

Specify 0.

(H8/300H)

RAPI_ADDRA	Selects A/D data register A.
RAPI_ADDRB	Selects A/D data register B.
RAPI_ADDRD	Selects A/D data register C.
RAPI_ADDRD	Selects A/D data register D.

Return value

If A/D converted value was successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

[__CreateADC](#), [__EnableADC](#), [__DestroyADC](#), [__GetADCAI](#)

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    unsigned int data;

    /* Get an A/D converted data of A/D register 0 */
    __GetADC( RAPI_AD0, &data );
}
```

__GetADCAI

Synopsis

<Get A/D converted value (all registers)>

Boolean **__GetADCAI(unsigned int *data)**

data	Pointer to the buffer in which A/D converted value is stored.
------	---

Description

Gets the A/D converted value from all A/D registers.

The A/D registers from which A/D converted values are acquired are listed below.

[M16C]	:	[0] A/D register 0
(16 bytes)		[1] A/D register 1
		[2] A/D register 2
		[3] A/D register 3
		[4] A/D register 4
		[5] A/D register 5
		[6] A/D register 6
		[7] A/D register 7

[R8C]	:	A/D register
(2 bytes)		

[H8/300H]	:	[0] A/D register A
(8 bytes)		[1] A/D register B
		[2] A/D register C
		[3] A/D register D

Return value

If A/D converted values were successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

[__CreateADC](#), [__EnableADC](#), [__DestroyADC](#), [__GetADC](#)

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    unsigned int data[8];

    /* Get A/D converted datas of A/D register */
    __GetADCAI( data );
}
```

__GetADCStatus

Synopsis

<Get the status of A/D converter>

Boolean **__GetADCStatus(unsigned int *status)**

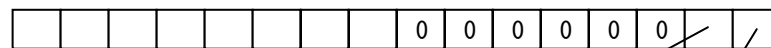
status	Pointer to the buffer in which the register content indicating A/D converter status is stored.
---------------	--

Description

Gets the status of a specified A/D converter.

The status of interrupt bit (when using the M16C or R8C) or the value of A/D end flag (when using the H8/300H) is stored in the first low-order bit of *status. Furthermore, the status of A/D conversion start flag is stored in the second low-order bit of *status. When used in the M16C, the value of A/D conversion status register 0 is stored in the 8 high-order bits of *status.

[Configuration of *status]



Value of A/D conversion status
Value of A/D conversion start flag (for only register 0 (for only the M16C; 0 for the R8C; 0 for the M16C and H8/300H)

Return value

If A/D converter status was successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

[__ClearADCStatus](#)

Program example

```
#include " rapi_ad_r8c_13.h"

void func( void )
{
    unsigned int status;

    /* Get status of A/D convertered */
    __GetADCStatus( &status );
}
```

__ClearADCStatus

Synopsis

<Clear the status of A/D converter>

Boolean __ClearADCStatus(unsigned int status)

status	Status of A/D converter
--------	-------------------------

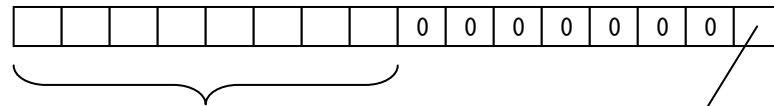
Description

Clears the status flag of a specified A/D converter.

Specify the status of interrupt bit (when using the M16C or R8C) or the value of A/D end flag (when using the H8/300H) in the first low-order bit of status.

When used in the M16C, specify the value of A/D conversion status register 0 in the 8 high-order bits of status. Write 0 to the bits to be cleared and 1 to the bits that do not need to be cleared.

[Configuration of status]



Value of A/D conversion status register 0 (for the M16C; 0 for the R8C and H8/300H) Value of interrupt bit or A/D end flag

Return value

If A/D converter status flag was successfully cleared, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

[__GetADCStatus](#)

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    /* Clear status of A/D convertered */
    __ClearADCStatus( 0 );
}
```


Renesas Embedded
Application Programming Interface
Reference Manual

Rev. 1.01
Issued: April 2007

Renesas Embedded
Application Programming Interface
Reference manual

